webMethods EntireX

Installation under z/OS

Version 9.5 SP1

November 2013

This document applies to webMethods EntireX Version 9.5 SP1.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: EXX-INSTALL-95SP1-20140628ZOS

Table of Contents

Installing EntireX under z/OS v
1 z/OS Prerequisites 1
2 Introduction to Installing EntireX under z/OS 5
Installation Method 6
Licensing Considerations 6
Installable Components 7
Contents of Mainframe Installation Medium 7
Installation Jobs 9
Storage Requirements 9
Copying the Contents of the Installation Medium to Disk 10
3 Simplified z/OS Installation Method 13
Overview 14
Delivered Members 14
Installation Parameters 17
Installation Jobs 20
Installation Verification 22
Administration Tasks 22
4 Installing EntireX Broker under z/OS 25
Step 1: Set up the Attribute File 26
Step 2: Edit the Broker Startup Procedure 26
Step 3: Install the Broker Stubs 27
Step 4: Authorize the Broker STEPLIB Data Sets 28
Step 5: Configure Time Zone Settings 28
Step 6: (Optional) Define the Persistent Store 29
Step 7: (Optional) Load the INPL and ERRN Files: Install the Natural-based EntireX Broker Tutorial 30
5 Installing the EntireX RPC Servers under z/OS 33
Installing EntireX RPC Servers under CICS 34
Installing EntireX RPC Servers under Batch 38
Extract the EntireX RPC Examples from their Container Data Set 40
6 Installing EntireX Security under z/OS 41
Installing EntireX Security for Broker Kernel 42
Setting up EntireX Security for Broker Stubs 45
7 Installing EntireX Java Components under z/OS UNIX 49
Scope 50
Installation Steps 50
Configuring and Administering the Java Components 51
8 Verifying the z/OS Installation 53
Testing the EntireX Broker Interface Installation 54
Verify the Installation of the EntireX RPC Server 54
9 Installing Adabas Components for EntireX under z/OS 57
Initializing the Adabas Communication Environment 58
SVC Integrity Validation 68

Requirements for Cross-Memory Services 69
Applying Zaps 70
10 Installing Adabas with TP Monitors 73
Preparing Adabas Link Routines for IBM Platforms 74
Installing Adabas with IMS/TM under Adabas 8 75
General Considerations for Installing Adabas with CICS 77
Installing Adabas with CICS under Adabas 8 79
Installing Adabas with Com-plete under Adabas 8 82
General Considerations for Installing Adabas with Batch/TSO 84
Installing Adabas with Batch/TSO under Adabas 8 86
Modifying Source Member Defaults (LGBLSET Macro) in Version 8 88

Installing EntireX under z/OS

There are two methods for installing EntireX under z/OS:

- the new, simplified installation method; this is the method we recommend
- the classic installation method as used in previous versions of EntireX

The information you need depends on the installation method you choose.

The following sections apply to both methods:	
Prerequisites	Prerequisites
Introduction	Installation scope; contents of the mainframe installation medium; copying the contents to disk.
The following section applies to the simplified method only:	
Simplified Installation	New, simplified installation procedure. All JCL members are now in library EXX951.JOBS. This data set has been thoroughly restructured; it contains installation, installation verification and maintenance jobs of all EntireX subproducts. This is the recommended installation method.
The following sections apply to the classic installation method only:	
Installing EntireX Broker under z/OS	Describes the steps for installing EntireX Broker under z/OS.
Installing EntireX RPC Servers	Describes how to install the EntireX RPC servers under CICS, Batch and IMS.
The following sections apply to both methods:	
Verifying the Installation	Describes how to verify whether the installation of EntireX Broker and EntireX Developer's Kit was successful.
Installing EntireX Security	Provides links to the steps required for installing EntireX Security under z/OS.
Installing EntireX Java Components	How to install EntireX Java components under z/OS UNIX.
Installing Adabas Components for EntireX	Installing the Adabas components for EntireX if you do not have Adabas already installed at your site and you want to use SVC communication.
Installing Adabas with TP Monitors for EntireX	Installing Adabas components for EntireX in batch mode and with its teleprocessing (TP) monitors.

Installation prerequisites are described for all platforms centrally. See *z/OS Prerequisites* in the EntireX Release Notes.

Note: If you want to use EntireX on z/OS together with the Eclipse-based EntireX Workbench components, you need to install the respective EntireX components under UNIX or Windows, using the Software AG Installer. See the separate Software AG Installer documentation

under *webMethods Product Documentation* > *webMethods Product Suite* > *Documentation by Product* > *System Requirements, Installation, and Upgrade* on the **Software AG Product Documentation** website.

1 z/OS Prerequisites

Note: z/OS 1.12 or higher is required for all components. For information regarding Software AG product compatibility with IBM platforms and any IBM requirements for Software AG products, see *Product Compatibility for IBM Platforms* on the Software AG website.

Component	Prerequisites
EntireX Broker	■ **Transport Options** ■ TCP-based communications: IBM TCP Stack ■ SSL-based communications: IBM GSK ■ NET-based communications: Entire Net-Work. See **note below**. ■ **EntireX Security** SAF-compatible security system for host z/OS compatible Broker kernel: ■ Resource classes/types and profiles as required by the installed security system. This can require a machine IPL in the case of RACF. ■ If you are using the trusted SAF user ID feature with the CICS TP monitor, set ADAGSET macro parameter `SAF=YES` when installing the Adabas/CICS link module.
CICS RPC Server Batch RPC Server IMS RPC Server	■ If applicable, see prerequisites for **COBOL Wrapper** and **PL/I Wrapper**. ■ Same prerequisites as **Broker Stubs**. ■ IBM Assembler for CICS RPC Server.
Broker Stubs	■ **Transport Options** See Broker prerequisites above.

Component	Prerequisites
	■ **Lowest Supported Applications Environment Versions** ■ For CICS applications: CICS TS 3.1 ■ For IMS-based applications: IMS 8 ■ For Com-plete based applications: Com-plete 6.5; stub COMETB requires APS331 SP5 ■ For Natural-based applications: NAT 4.2.6
Workbench	■ **COBOL Wrapper** ■ To compile the sources generated by the EntireX Workbench component COBOL Wrapper: Compiler supported by the COBOL Wrapper: standard COBOL compiler, for example IBM Enterprise COBOL for z/OS 4.2. ■ For client side, see prerequisites for **Broker stubs**. ■ For server side, an RPC server. See prerequisites for relevant RPC server above. ■ **PL/I Wrapper** ■ To compile the sources generated by the EntireX Workbench component PL/I Wrapper: Compiler supported by the PL/I Wrapper, for example PL/I for MVS & VM V1R1.1, or the newer Enterprise PL/I for z/OS V4.R1. ■ For client side, see prerequisites for **Broker stubs**. ■ For server side, an RPC server. See prerequisites for relevant RPC server above. ■ **IDL Extractor for Natural** To extract from z/OS, a standard Natural RPC server is required with one of the following Natural versions. The scope of the generation depends on the version: ■ Natural 4.2.7.4 is required for improved handling if IDL extractions fails. ■ Natural 4.2.7 or higher is required to make use of the following features: ■ during runtime to support a redesigned interface including support for `REDEFINE`s, map to suppress etc. ■ Optional replacing of special characters ("@") in parameter names by underscore ■ Natural 4.2.6.2 or higher is required for object extractions to make use of the features (introduced with EntireX 8.1 SP2). ■ Optional replacing of special characters ("#", "$", "&", "/") in parameter names by underscore ■ Hints (comments) in extracted IDL per parameter for restrictions and usage where appropriate. ■ Support for Natural V-arrays (Natural syntax: A100/1:V). They are mapped to IDL unbounded arrays. ■ Combinations of Natural X-array dimensions together with Natural V-array dimensions (Natural syntax: A100/1:*,1:V) are supported. They are mapped to IDL unbounded arrays.

Component	Prerequisites
	■ Natural 4.2.6.1 or lower can be used, but if extracted from objects there is no support of the features supported by Natural 4.2.6.2 (see above). ■ **Natural Wrapper** To generate a client interface object, a standard Natural RPC server is required with one of the following Natural versions. The scope of the generation depends on the version: ■ Natural 4.2.7.11 or higher or Natural 8.2.2.6 or higher is required to disable/enable the generation of Natural client test programs. See *Step 4a: Customize Natural Client Names and Save Remotely (Optional)* | *Step 4b: Customize Natural Client Names and Save Locally (Optional)*. ■ Natural 4.2.7.8 or higher or Natural 8.2.2.2 or higher is required to adapt the names for the Natural client interface objects (subprograms (NSNs) with their parameter data areas (PDAs)). See See *Step 4a: Customize Natural Client Names and Save Remotely (Optional)* | *Step 4b: Customize Natural Client Names and Save Locally (Optional)*. ■ Natural 4.2.7.4 or higher is required if a *CVM File* is used in addition to an IDL file to support features introduced with EntireX 8.2 such as multiple Natural subprogram interfaces, `REDEFINE`s or map to suppress. See *Redesigning the Extracted Interface* in the IDL Extractor for Natural documentation. ■ Natural 4.2.6.2 or higher generates Natural client interface objects, separate Natural parameter data areas (PDAs) and Natural client test programs. ■ Natural 4.2.5.5 to Natural 4.2.6.1 generates Natural client interface objects only. To generate a server skeleton, a standard Natural RPC server is required with Natural version 4.2.7.11 or higher or Natural 8.2.2.6 or higher. See *Using the Natural Wrapper for the Server Side*.
Attach Manager	■ **Transport Options** ■ TCP-based communications: IBM TCP Stack ■ NET-based communications: Entire Net-Work. See **note below**. The attach services supplied with Broker Services are still supported in this version but may be replaced in the future.

Additional Notes for z/OS

■ **Entire Net-Work**

- EntireX works with any supported version of Entire Net-Work. We recommend you use the latest version, which for z/OS is currently 6.3. The Adabas version we recommend is 8.2.
- Adabas Cross-Memory Services are required if you are using NET transport or using Adabas as your persistent store. We recommend using at least the version delivered with EntireX (WAL825). See ***Installing Adabas Components for EntireX under z/OS***.

2 Introduction to Installing EntireX under z/OS

- Installation Method 6
- Licensing Considerations 6
- Installable Components 7
- Contents of Mainframe Installation Medium 7
- Installation Jobs 9
- Storage Requirements 9
- Copying the Contents of the Installation Medium to Disk 10

Installation Method

There are two methods for installing EntireX under z/OS:

- the new, simplified installation method
- the classic installation method as used in previous versions of EntireX

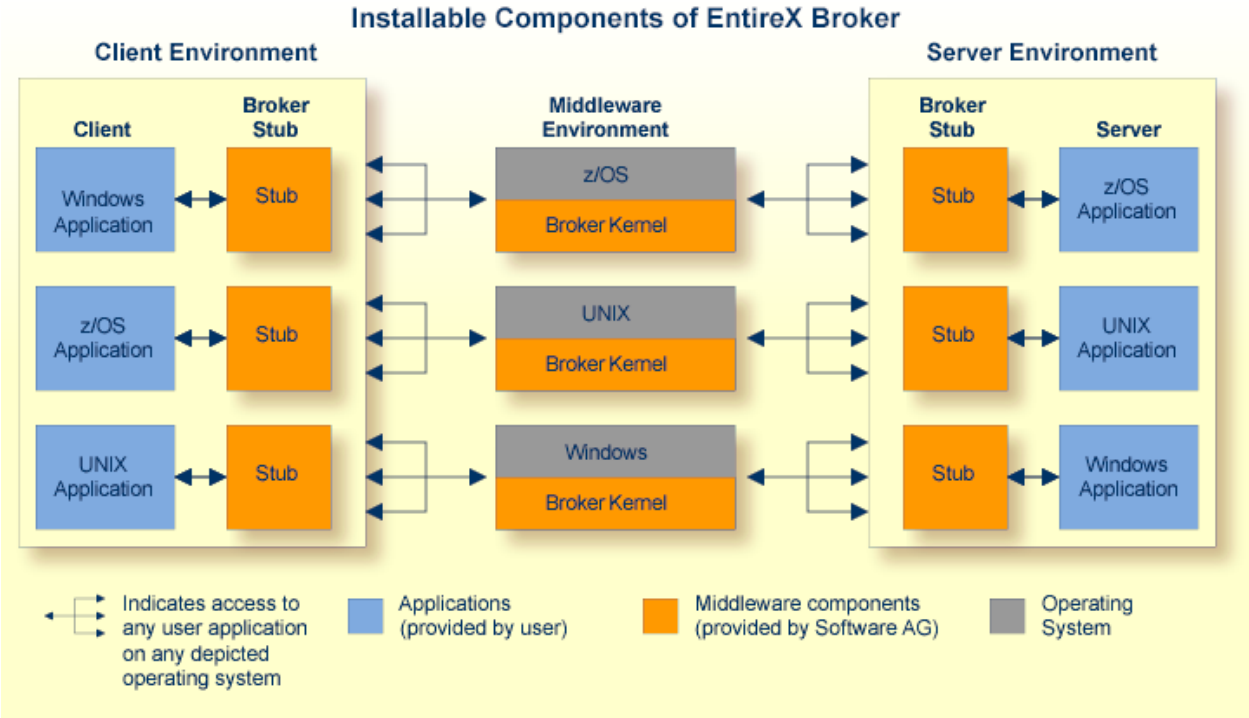
The information provided in this section applies to both methods. Then continue with either ***Simplified z/OS Installation Method*** or ***Installing EntireX Broker under z/OS***.

Licensing Considerations

Software AG licensing requires that the modules `LICMAIN` and `LICUTIL` can be loaded when the EntireX Broker starts up. These modules are distributed in the MLC*vrs*.LOAD library.

See *z/OS Licensing Data Sets* under *EntireX Installation Overview* in the general installation documentation.

Installable Components



Contents of Mainframe Installation Medium

The webMethods EntireX installation medium contains the data sets required to install all EntireX z/OS components. Data set names begin with a product code that identifies the module, as in the following tables.

With webMethods EntireX, the following additional products are included:

- Various Base Products for EBCDIC platforms
- Transport Services for EBCDIC platforms

Code	webMethods EntireX Component
EBV	EntireX Broker Services (see *Attach Manager* under under *z/OS Prerequisites* in the EntireX Release Notes).
EXP	EntireX RPC (mainframe part of EntireX Developer's Kit).
EXB	EntireX Broker.
EXX	Common modules used by all EntireX components.

Code	webMethods EntireX Component
IAF	Integrated Authentication Framework.
MLC	Software AG's common mainframe license check software.
SSX	Software AG Security Extensions.
WAL	Adabas Limited Libraries.

The installation medium contains the data sets listed below:

Data Set Name	Description
SMT111.TABS	SMA tables and jobs.
EBV*vrs*.ERRN	Broker Services error message library.
EBV*vrs*.INPL	Broker Services-specific INPL library.
EBV*vrs*.LOAD	Broker Services-specific load library.
EBV*vrs*.SASC	Broker Services runtime for z/OS.
EBV*vrs*.SRCE	Broker Services-specific source library.
EBV*vrs*.SYSF	Broker Services-specific INPL library.
EBV*vrs*.ZAPS	Broker Services zaps library.
EXB951.ERRN	Broker Natural tutorial error message data set.
EXB951.INPL	Broker Natural tutorial INPL data set.
EXB951.LOAD	Broker-specific load library (PDS/E).
EXB951.SRCE	Broker-specific source library
EXP951.EXPL	EntireX Developer's Kit RPC compressed examples library (COBOL and PL/I).
EXP951.INCL	Developer's Kit RPC include and copybook library.
EXP951.LB00	Developer's Kit RPC batch load library.
EXP951.LD00	Developer's Kit RPC CICS load library.
EXP951.MACS	Developer's Kit RPC macros.
EXP951.SD00	Developer's Kit RPC side deck library.
EXP951.SRCE	Developer's Kit RPC source data set.
EXX951.CERT	Certificates to demonstrate SSL/TLS connections.
EXX951.DC00	Readme.
EXX951.LICS	License key.
EXX951.JOBS	This data set has been thoroughly restructured. It contains installation, installation verification and maintenance jobs of all EntireX subproducts.
EXX951.LOAD	Common load library (PDS/E), also includes broker stubs.
EXX951.SD00	Common side deck library.
EXX951.TAR	All Java components for z/OS. See ***Installing EntireX Java Components under z/OS UNIX***.
EXX951.SRCE	Common source library.
EXX951.ZAPS	Common zaps library.

Data Set Name	Description
IAF*vrs*.JOBS	IAF-specific jobs library.
IAF*vrs*.LOAD	IAF-specific load library.
IAF*vrs*.SRCE	IAF-specific source library.
MLC*vrs*.JOBS	Sample job library for Software AG's common mainframe license check software.
MLC*vrs*.LOAD	Load library for Software AG's common mainframe license check software.
SSX*vrs*.LOAD	SSX-specific load library.
WAL825.LOAD	Adabas limited load library.
WAL825.SRCE	Adabas limited source library.
WAL825.JOBS	Adabas limited jobs library.

Note: In the table above, *vrs* represents the version, release and service pack. Make sure you install the highest patch level available.

Installation Jobs

The installation of Software AG products on z/OS is performed by installation jobs. These jobs are either created manually or generated by System Maintenance Aid (SMA).

For each step of the installation procedure, an installation job is generated by SMA according to your specifications in SMA. If you are not using SMA, follow the instructions using your own jobs.

Information on using SMA for the installation process is provided in the *System Maintenance Aid Manual*.

Storage Requirements

For specific storage requirements, see the Software AG Product Delivery Report.

Copying the Contents of the Installation Medium to Disk

Installation Steps

Copy the data sets from the supplied installation medium to your disk before you perform the individual installation procedure for each component to be installed.

The way you copy the data sets depends on the installation method and the medium used:

- If you use System Maintenance Aid (SMA), refer to the copy job instructions provided in the *System Maintenance Aid* documentation.
- If you are not using SMA and want to copy the data sets from CD-ROM, refer to the README.TXT file on the CD-ROM.
- If you are not using SMA and want to copy the data sets from tape, follow the instructions in this section.

This section explains how to copy all data sets from tape to disk.

- Step 1: Copy Data Set COPY.JOB from Tape to Disk
- Step 2: Modify hilev.COPY.JOB on Your Disk
- Step 3: Submit COPY.JOB

Step 1: Copy Data Set COPY.JOB from Tape to Disk

- Modify the following sample job according to your requirements:

```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
//* --------------------------------
//COPY EXEC PGM=IEBGENER
//SYSUT1 DD DSN=COPY.JOB,
// DISP=(OLD,PASS),
// UNIT=(CASS,,DEFER),
// VOL=(,RETAIN,SER=tape-volser),
// LABEL=(2,SL)
//SYSUT2 DD DSN=hilev.COPY.JOB,
// DISP=(NEW,CATLG,DELETE),
// UNIT=3390,VOL=SER=disk-volser,
// SPACE=(TRK,(1,1),RLSE),
// DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//
```

where:

tape-volser is the VOLSER of the tape, for example: `T12345`,

hilev is a valid high-level qualifier, and
disk-volser is the VOLSER of the disk.

- Execute the job to copy the data set COPY.JOB to your disk.

Step 2: Modify hilev.COPY.JOB on Your Disk

- Modify *hilev*.COPY.JOB according to your requirements:

  Set EXPDT to a valid expiration date, for example, 99365.

  Set HILEV to a valid high-level qualifier, for example, USERLIB.

  Set LOCATION to a storage location, for example, STORCLAS=ABC or UNIT=3390,VOL=SER=USR123.

Step 3: Submit COPY.JOB

- Execute *hilev*.COPY.JOB to copy single, multiple, or all data sets to your disk.

Next Steps

Continue the installation with either ***Simplified z/OS Installation Method*** or ***Installing EntireX Broker under z/OS***.

3 Simplified z/OS Installation Method

- Overview 14
- Delivered Members 14
- Installation Parameters 17
- Installation Jobs 20
- Installation Verification 22
- Administration Tasks 22

Overview

The simplified installation requires the following steps:

1. Set the installation keyword parameters in a parameter member.
2. Execute a REXX script that updates the delivered installation jobs with the values of that parameter member.
3. Auto-submit the updated jobs if requested in the parameter file, or make them ready for a manual submission.

Delivered Members

The delivered members in the original `EXX951.JOBS` data set are:

`#IPARMS`
: The installation parameters.

`#RX#CMD`
: The REXX update script.

`#RX#JOB`
: A job to execute `#RX#CMD` in batch.

`#INSTALL`
: Generated on first submission: Documents the installation jobs required as a result of the components selected.

Note: Only the `#INSTALL` job will be copied to the target *.JOBS* data set. The first three members remain in the original *EXX951.JOBS* data set.

#IPARMS - The Parameters

This member keeps all parameters relevant for installation and is located and maintained in the delivered data set *EXX951.JOBS* only, which was copied directly from the installation medium. To avoid a complete setup, you can select an already maintained #IPARMS. See *#RX#JOB - A REXX Batch Job*.

Member Layout

The parameter member contains the following sections:

1. global installation parameters
2. job card related parameters
3. installable unit subproduct selection parameters
4. installable unit installation parameters
5. data set names / high-level qualifiers
6. VSAM file characteristics

Parameter Line Layout

A parameter member line has the following layout:

```
* ---------------------------------------------------------------- *
* JOBCARD
* ---------------------------------------------------------------- *
  keyword   = value           * comment
```

where *keyword* is a placeholder for a keyword

= is the value separator

value is the keyword value

* introduces a comment. A line that begins with an asterisk in column 0 is treated as a comment line.

#RX#CMD - The REXX Script

This script copies all the necessary jobs from the original *EXX951.JOBS* data set of the delivered installation medium to a selected installation *target.JOBS* data set. All placeholders in the original jobs will be replaced with the values set in the parameter member. An existing *target.JOBS* data set of a previous generation will be saved in a GDG data set first.

A #INSTALL member will be created for documentation and to distinguish between the jobs required as a result of the selected components and the optional installation jobs. All required installation jobs can be submitted directly with this member.

#RX#JOB - A REXX Batch Job

Calls the #RX#CMD `REXX` script from batch.

The #RX#CMD script expects six KEYWORD parameters that need to be maintained and set to suitable values in the #RX#JOB first. There are two groups of keyword parameters: mandatory and optional.

- **Mandatory Keyword Parameters**
  - ORIG - The name of the EXX951.JOBS data set copied from the installation medium
  - INST - The name of the target.JOBS data set that will contain the generated installation jobs
- **Optional Keyword Parameters**
  - PARM - The name of a data set containing an already maintained #IPARMS member. If a #IPARMS member is found here it will be selected for the installation jobs generation process.
  - MSGS - Switch IEBCOPY sysout messages on or off. Possible values are:
    Y - Show IEBCOPY sysout (default)
    N - Hide IEBCOPY sysout
  - INST_SMS - Switch SMS control on or off for the target.JOBS data set named with the mandatory INST keyword parameter. Possible values:
    Y - Switch ON the SMS controlled allocation of the installation jobs data set
    N - Switch OFF the SMS controlled allocation of the installation jobs data set
  - INST_VOL - Contains the SMS class name or the VOLSER name depending on the value of keyword parameter INST_SMS
    INST_SMS=Y - The SMS class name
    INST_SMS=N - The DASD VOLSER name

Important: It is mandatory to edit this job *manually* before submission to define at least the mandatory keywords naming the input (*EXX951.JOBS*) and output (*`target`.JOBS*) data sets.

#INSTALL - Document the Required Installation Jobs (Depending on Selected Components)

This member will be generated in both, the *EXX951.JOBS* and the *`target`.JOBS* data sets. It has two functions:

1. **Documentation of the Installation Jobs Required as a Result of the Selected Installable Units**
   Each required installation job is represented here with a separate line headed by its member name in the *EXX951.JOBS*.
2. **Submission of the Required Installation Jobs**
   As this member is a job itself, it can be submitted to submit all the required installation jobs. You can also select a subset of jobs for submission by (un)commenting their execution lines.

Installation Parameters

This section lists the parameters available with the simplified installation procedure:

- Installable Unit Selection Parameters
- Job Replacement Parameters

Installable Unit Selection Parameters

Use these parameters to determine the scope of the installation.

Installable Unit	Description
InstBroker	Subproduct installation: Select the Broker (Y/N).
InstRpcBatch	Subproduct installation: Select the Batch RPC server (Y/N).
InstRpcCics	Subproduct installation: Select the CICS RPC Server (Y/N).
InstRpcIms	Subproduct installation: Select the IMS RPC server (Y/N).
InstBrokerServices	Subproduct installation: Select the Broker Services (Y/N).
InstSSL	Subproduct installation: Select the SSL libraries (Y/N).

Job Replacement Parameters

Parameter Type	Parameter Name	Job Replacement Tag	Description
Global Parameters	WorkingStorUnit	<unit>	Temporary storage unit.
	MaxBackups	<MaxBackups>	Upper limit of GDG data set.
	HLQ-TapeKit	<Tape>	High-level qualifier of the JOBS data sets copied to DASD from the delivered installation medium.
	HLQ-InstallKit	<Inst>	High-level qualifier of the data set containing the generated installation jobs.
	Submit	<,>	Autosubmit (Y/N) the mandatory installation jobs for the selected units to the HOLD queue.
Job Card Related	JobName	<jobname>	Job name.
	Accounting	<account>	Accounting information.
	Programmer	<name>	Programmer information.
	Notify	<notify>	Notification user ID.
	Region	<region>	Storage amount.
	Priority	<prio>	Selection priority.
	Time	<time>	Max. CPU time.

Parameter Type	Parameter Name	Job Replacement Tag	Description
	MsgLevel	<msglvl>	Job output level.
	JobClass	<class>	Run class.
	MsgClass	<msgclass>	Message class.
	Special	<special>	Special job info (e.g. for JES exits).
High-level Qualifiers	HLQ-Adabas	<ADAvrs>	High-level qualifier of the installed Adabas.
	HLQ-AdabasIndi	<WALvrs>	High-level qualifier of the installed Independent Adabas.
	HLQ-AdabasDB	<ADAdb>	High-level qualifier of the target Adabas database for the ADA/NAT update jobs.
	HLQ-Natural	<NATvrs>	High-level qualifier of the installed Natural.
	HLQ-EntireX	<EXXvrs>	High-level qualifier of the installed EntireX Global.
	HLQ-Broker	<EXBvrs>	High-level qualifier of the installed EntireX Broker.
	HLQ-RpcServer	<EXPvrs>	High-level qualifier of the installed EntireX RPC Servers.
	HLQ-SSX	<SSXvrs>	High-level qualifier of the installed EntireX SSX.
	HLQ-BrokerServices	<EBVvrs>	High-level qualifier of the installed EntireX Broker Services.
	HLQ-Cics	<CICS>	High-level qualifier of the CICS to be used for installation.
	HLQ-CicsCSD	<CICSCSD>	High-level qualifier of the CICS CSD to be used for installation.
	HLQ-IMS	<IMS>	High-level qualifier of the IMS to be used for installation.
	HLQ-MQM	<MQM>	High-level qualifier of the MQSeries data sets.
	HLQ-COB	<COB>	High-level qualifier of the COBOL compiler data sets.
	HLQ-PLI	<PLI>	High-level qualifier of the PL/I compiler data sets.
	HLQ-CEE	<CEE>	High-level qualifier of the CEE language environment data sets.
Broker-related	BrokerID	<BrokerID>	EntireX Broker name for TCP/IP or Entire Net-Work access.
	BrokerNode	<BrokerNode>	EntireX Broker node number for Entire Net-Work access.
	MigDsnBrokerV72	<EXBmig>	Broker source data set for automated parameter migration.

Parameter Type	Parameter Name	Job Replacement Tag	Description
	MigDsnEntireXV72	<EXXmig>	EntireX source data set for automated parameter migration.
	MigOldAttrMember	<OldAttrMember>	Migration: Member name of old attributes (EXBATTR).
	MigOldCommMember	<OldCommMember>	Migration: Member name of old TCP/IP parameters (EXBCOMM).
	MigOldStorMember	<OldStorMember>	Migration: Member name of old persistent storage parameters (PSFFP).
	MigOldParmMember	<OldParmMember>	Migration: Member name of old parameters (EXBPARM).
	MigNewAttrMember	<NewAttrMember>	Migration: Member name of new consolidated attributes.
RPC Server-related	IMS-PsbName	<ImsPsbName>	IMS RPC: PSB name.
	DSN-ImsLoad3GL	<DSNImsLoad3GL>	IMS RPC: COBOL and PL/I server load modules data set.
	DSN-BatLoad3GL	<DSNBatLoad3GL>	Batch RPC: COBOL and PL/I server load modules data set.
	DSN-DfhLoad3GL	<DSNDfhLoad3GL>	CICS RPC: COBOL and PLI/I server load modules data set.
	DSN-ImsStub3GL	<DSNImsStub3GL>	IMS RPC: COBOL and PL/I stub load modules data set.
	DSN-BatStub3GL	<DSNBatStub3GL>	Batch RPC: COBOL and PL/I stub load modules data set.
	DSN-DfhStub3GL	<DSNDfhStub3GL>	CICS RPC: COBOL and PL/I stub load modules data set.
	HLQ-RpcCicsSVM	<HlqDfhSvmFile>	High-level qualifier of the CICS EntireX RPC server mapping (SVM) file.
	HLQ-RpcSVM	<HlqSvmFile>	High-level qualifier of the Batch/IMS EntireX RPC SVM file.
	RPC-ImsClass	<ImsClass>	IMS RPC: Class name of service triplet.
	RPC-ImsServer	<ImsServer>	IMS RPC: Server name of service triplet.
	RPC-ImsService	<ImsService>	IMS RPC: Service name of service triplet.
Adabas / Natural-related Parameters	AdaSvcNo	<AdaSvcNo>	Adabas router SVC number.
	AdaDeviceType	<AdaDeviceType>	Adabas database device type.
	AdaDBID	<AdaDBID>	Adabas database ID in the router SVC.
	NatBatch	<NatBatch>	Name of the Natural batch nucleus.
	Fnat	<Fnat>	Adabas file number of the Natural FNAT.
	Fuser	<Fuser>	Adabas file number of the Natural FUSER.
	Fdic	<Fdic>	Adabas file number of the Natural FDIC.

Parameter Type	Parameter Name	Job Replacement Tag	Description
	AdaFileNoEbvLog	<LogFileNo>	Adabas file number of the Broker Services log.
	AdaFileNoExbDiv	<DivFileNo>	Adabas file number of the Broker persistent store.
SVM VSAM file allocation characteristics	RLS-SVM	<rls>	Record Level Sharing (Yes or No).
	SMS-SVM		SVM managed (Yes or No).
	VolSer-SVM	<sms>	SMS storage class name or VOLSER name (depending on SMS-SVM).
Broker VSAM file allocation characteristics for persistent store linear cluster	LLQ-EXB	<llq-exb>	Last-level qualifier of the Broker VSAM cluster.
	SMS-EXB		SVM managed (Yes or No).
	VolSer-EXB	<sms-exb>	SMS storage class name or VOLSER name (depending on SMS-SVM).

Installation Jobs

All existing jobs remain unchanged in the original *EXX951.JOBS* data set. Only the jobs necessary for the installation of the selected installable units will be copied to and updated within the designated *target.JOBS* data set by the `#RX#CMD` REXX script.

Note: This update process is not mandatory. All jobs can still be manually adapted as before. But to be able to successfully execute the installation script afterwards, the original *EXX951.JOBS* data set should be kept unchanged.

This section covers the following topics:

- Generation Data Sets
- Preparation and Execution
- Generation Process

- Submission

Generation Data Sets

After a successful generation, the following data sets are in the system:

- ***EXX951.JOBS***
  The delivered product data set. The members of this data set will remain unchanged with the exception of the #IPARMS parameters and possibly the #RX#JOB.
- ***target.JOBS***
  The target data set which will contain the selected and updated jobs from the *EXX951.JOBS* data set.
- ***target.JOBS.BAK***
  A VSAM GDG base catalog entry for the GDG data sets.
- ***target.JOBS.****GnnnnVnn*
  With any subsequent generation, all jobs of the previous generation in the *target.JOBS* data set are kept here until the maximum value set in the MAXBACKUPS parameter (to be found in #IPARMS) is reached.

Preparation and Execution

To prepare and install the installation jobs

Note: It is mandatory to edit the job *manually* first to set the input and output JOBS data set names.

1 Update the *EXX951.JOBS* (#IPARMS) parameter member.

2 Submit the job EXX951.JOBS(#RX#JOB) to execute the REXX script #RX#CMD.

Generation Process

Every keyword parameter has a well defined default value. This default will be replaced only if requested by the parameter member, that is, if a KEYWORD=VALUE entry is successfully identified. When a keyword is deleted from the parameter member (or commented out), this default will always be in place.

Submission

To submit the installation jobs

- In the #IPARMS member select the switch SUBMIT.
  - If set to "Y":

    Any job is submitted with a TYPRUN=HOLD job card parameter.

    Note: This allows a final check to ensure a correct generation. Any job can then be released manually one after the other in the numbered order or can be cancelled if an error was encountered.

    If you cannot issue the JES release command for security reasons, you can submit using the generated #INSTALL job in the target.JOBS instead.
  - If set to "N":

    The TYPRUN=HOLD job card parameter is set to comment and no job is submitted at all.

Note: The #INSTALL member will be generated regardless of the SUBMIT parameter value.

Installation Verification

Installation verification is the same for both installation methods. See *Verifying the z/OS Installation*.

Administration Tasks

After installation has been completed, various administration tasks may be necessary.

- Modify Broker Attribute File
- Set up Broker Stubs
- Define the Persistent Store

- Setting Up the EntireX RPC Servers

Modify Broker Attribute File

Customize the attribute settings to suit your needs. See *Broker Attributes* in the administration documentation.

Set up Broker Stubs

See *Administration of Broker Stubs under z/OS* in the z/OS administration documentation.

Define the Persistent Store

A persistent store can be optionally used for storing unit of work messages and message status information to disk. For z/OS, you can use an Adabas persistent store (recommended) or a DIV persistent store that uses a VSAM linear data set.

Adabas Persistent Store

See *Implementing an Adabas Database as Persistent Store* under *Managing the Broker Persistent Store* in the z/OS administration documentation and *Adabas-specific Attributes* (`DEFAULTS=ADABAS`) under *Broker Attributes* in the administration documentation in the Broker attribute file documentation for more information.

DIV Persistent Store

See *Implementing a DIV Persistent Store* under *Managing the Broker Persistent Store* in the z/OS administration documentation and *DIV-specific Attributes* (`DEFAULTS=DIV`) under *Broker Attributes* in the administration documentation for more information.

Setting Up the EntireX RPC Servers

See *CICS RPC Server*, *Batch RPC Server* or *IMS RPC Server* in the z/OS administration documentation for more information.

4 Installing EntireX Broker under z/OS

- Step 1: Set up the Attribute File 26
- Step 2: Edit the Broker Startup Procedure 26
- Step 3: Install the Broker Stubs 27
- Step 4: Authorize the Broker STEPLIB Data Sets 28
- Step 5: Configure Time Zone Settings 28
- Step 6: (Optional) Define the Persistent Store 29
- Step 7: (Optional) Load the INPL and ERRN Files: Install the Natural-based EntireX Broker Tutorial 30

This chapter describes the steps for installing EntireX Broker under z/OS.

Step 1: Set up the Attribute File

(SMA Job I070 / Step 7604)

The EXB951.SRCE data set contains a sample Broker attribute file (member EXBATTR). Copy this member to a member of your choice and customize the attribute settings to suit your needs. The file must then be allocated to DD name ETBFILE in the Broker Startup Procedure (see step ***Step 2: Edit the Broker Startup Procedure***). For detailed information about the attributes, see *Broker Attributes* in the administration documentation.

Step 2: Edit the Broker Startup Procedure

(SMA Job I070 / Step 7606)

Copy the example member EXBSTART (the EntireX Broker Startup Procedure) from the EXX951.JOBS data set to your proclib and edit it to suit your naming conventions.

Note on Region Size

We recommend setting the region size for the Broker to "0M". This means the operating system allocates resources as required. If you wish to influence the resources allocated, you can also set the size depending on the amount of fixed virtual memory that is allocated during Broker initialization. This value is displayed in diagnostic message ETBD0284 of the Broker trace output. Add to this value an additional 64M to account for other storage possibly needed during the Broker execution.

Overview of DD Names

The following table describes the DD names used in this file:

DD Name	Description	See also Step
CEEDUMP	Exception handling will write any dump data to this data set.	
DDCLOGR1	First dual command log data set.	*Command Logging in EntireX* in the z/OS administration documentation.
DDCLOGR2	Second dual command log data set.	*Command Logging in EntireX* in the z/OS administration documentation.
ETBCREP	Configuration report provides the contents of the variable definition file (optional) and the contents of the attribute file (required).	
ETBFILE	Broker attribute file.	***Step 1: Set up the Attribute File***.
ETBLREP	License report provides the contents of the license file and some machine data.	
ETBMREP	Module report provides a list of all members in the STEPLIB data set concatenation.	
ETBPREP	Persistent store report provides a list of all records added to or deleted from the persistent store.	
ETBVARS	EXBVARS file.	***Step 1: Set up the Attribute File***.
LICENSE	EntireX License Certificate File.	
STORE01	Persistent store data sets file. Defines the DIV persistent store data set. This DD name is derived from the value specified in the persistent store format parameter `DDNAME`.	***Step 6: (Optional) Define the Persistent Store***.
ABNLIGNR	Needed to suppress ABEND-AID™ dumps.	

Step 3: Install the Broker Stubs

If you will be using the Broker stubs on z/OS, see *Administration of Broker Stubs under z/OS* in the z/OS administration documentation.

Step 4: Authorize the Broker STEPLIB Data Sets

The Broker DIV persistent store facility and the NET transport require that all data sets in the STEPLIB DD be APF-authorized. If one of these data sets is not APF-authorized, the network will not initialize, and will result in the following error message:

```
ETBE0090 EntireX Broker is not APF-authorized
```

A standard Broker will have the following data sets in the STEPLIB concatenation:

- EXX951.LOAD
- EXB951.LOAD
- WAL825.LOAD

Step 5: Configure Time Zone Settings

Perform the following to configure time zone settings.

- Step 5a: Give Broker Access to the /etc/profile File
- Step 5b: Verify Correct Time Zone Setting for Broker

Step 5a: Give Broker Access to the /etc/profile File

Ensure that your security product (e.g., RACF, ACF/2) permits the Broker kernel to have read access to the */etc/profile* file.

The Broker kernel obtains the time zone setting from the TZ environment variable in the */etc/profile* file, if possible.

Note: If the Broker kernel is not allowed read access, daylight savings time (if applicable) will not be automatically recognized in the Broker kernel log messages. A message from the security product will indicate the lack of permission. Example:

```
ICH408I USER(QEUSALT1) GROUP(QADEPT  ) NAME(QEUSALT1 STC        )
  /etc/profile CL(FSOBJ   ) FID(01E2E8E2D9C4F500171E000001170000)
  INSUFFICIENT AUTHORITY TO OPEN
  ACCESS INTENT(R--)  ACCESS ALLOWED(OTHER      ---)
  EFFECTIVE UID(0000999999)  EFFECTIVE GID(0000000006)
```

Step 5b: Verify Correct Time Zone Setting for Broker

Verify that the time zone setting in the TZ environment variable of the */etc/profile* file is correct for your site, since the Broker obtains the time zone setting from this file, if possible.

Step 6: (Optional) Define the Persistent Store

The persistent store can be optionally used for storing unit of work messages and message status information to disk.

Important: For z/OS, the available persistent store types are Adabas file (perform step "a" below) and DIV file implemented with a VSAM linear data set (perform step "b" below).

See *Managing the Broker Persistent Store* in the z/OS administration documentation to understand how the persistent store is implemented.

Step 6a: Define and Migrate a New Adabas Persistent Store

Define the Adabas Persistent Store

(SMA Job I050 / Steps 7600 / 7610)

Use this step if you have not already defined an Adabas persistent store.

The Adabas persistent store driver module is contained within the regular Broker load library or binaries directory. Module ADAPSI is activated by specifying the `PSTORE-TYPE` parameter. Use the supplied job EXBJ015 from data set EXX951.JOBS to define and install the persistent store file in your Adabas database. This job creates and loads the Adabas file into the database.

See *Implementing an Adabas Database as Persistent Store* under *Managing the Broker Persistent Store* in the z/OS administration documentation and *Adabas-specific Attributes* (`DEFAULTS=ADABAS`) under *Broker Attributes* in the administration documentation for more information.

Migrate an Existing Adabas Persistent Store

Note: Migration of Adabas Persistent Store applies only when migrating from Broker 7.1 to Broker 8.0.

(SMA Job I051 / Step 7610)

If you already have an Adabas PSTORE file and want to use publish and subscribe functionality, use job SAGJ016 from data set EXX951.JOBS to modify the file so that it has the required format (FDT). This operation does not require you to perform a COLD start of Broker or to refresh the file.

See *Implementing an Adabas Database as Persistent Store* under *Managing the Broker Persistent Store* in the z/OS administration documentation for more information.

Step 6b: Allocate a DIV Persistent Store

To allocate the persistent store, you will need to define a VSAM linear data set (LDS) using the IBM utility IDCAMS. See *Implementing a DIV Persistent Store* under *Managing the Broker Persistent Store* in the z/OS administration documentation for details on creating the persistent store data sets.

The file containing the persistent store data set must then be allocated to DD name STORE01 in the Broker Startup Procedure (see Step ***Step 2: Edit the Broker Startup Procedure***).

Use the supplied job EXBIDCAMS from data set EXX951.JOBS to define a VSAM linear data set for the persistent store.

Note: You must allocate a unique persistent store (if used) per Broker.

See *Implementing a DIV Persistent Store* under *Managing the Broker Persistent Store* in the z/OS administration documentation for more information.

See also *DIV-specific Attributes* (`DEFAULTS=DIV`) under *Broker Attributes* in the administration documentation.

Step 7: (Optional) Load the INPL and ERRN Files: Install the Natural-based EntireX Broker Tutorial

(SMA Job I061 / Steps 7600, 7602)

Note: Perform this step only if you want to install the sample Natural programs.

1. Use the Natural system command `INPL` and `ERRLODUS` (see Natural User's Guide) to load the EntireX Broker system objects (EXB951.INPL and EXB951.ERRN).

   This loads the following library:

Library	File	Contents
SYSETB	FNAT	Sample Natural programs.

2. Set the Natural profile parameter ESIZE=40
3. Invoke Natural, logon to library SYSETB and edit the member PARM on library SYSETB to set parameters as required at your site, especially the BROKER-ID.

Parameter members can also be made user-dependent. Copy the PARM member and save it in a member with a user ID as name (Natural variable `*USER`). If a user logs on with this user ID, these parameters take effect instead of the PARM member.

5 Installing the EntireX RPC Servers under z/OS

- Installing EntireX RPC Servers under CICS 34
- Installing EntireX RPC Servers under Batch 38
- Extract the EntireX RPC Examples from their Container Data Set 40

Installing EntireX RPC Servers under CICS

The EntireX z/OS CICS® RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/OS under CICS. It supports the programming languages COBOL and PL/I.

- Update the CICS Tables
- Modify the CICS Startup JCL
- Build the ERXMAIN Control Block
- TCP/IP-enable the CICS Region (Optional)
- Install the SVM File for a CICS RPC Server (Optional)
- Starting the EntireX RPC Server Automatically on CICS Startup (Optional)
- Stopping the EntireX RPC Server Automatically on CICS Shutdown (Optional)
- Installing Multiple EntireX RPC Servers in the same CICS (Optional)
- Using SSL or TLS Connections with the CICS RPC Server (Optional)

Prerequisites for all EntireX components are described centrally. See *z/OS Prerequisites* in the EntireX Release Notes.

Update the CICS Tables

EntireX RPC requires a number of enhancements to the CICS tables. These changes are usually performed by a CICS system programmer using standard system jobs.

- Execute "Step 3: Install CICS CSD defs for EntireX RPC Server" of job EXPINSTA located in the EXX951.JOBS data set. This will add the group ERX with all its relevant entries.
- Tracing: TD queue entries for RPC server trace output may optionally be updated by uncommenting the DFHTRACE exec line. See also `ERXMAIN` *Macro* parameter `TRC1`.

Note: If you have not extended your DCT (that is, you are using the default values), you must specify the option DCT= in the CICS SYSIN file.

Modify the CICS Startup JCL

The startup JCL for CICS must be modified to include the EXP951.LD00 data set in the DFHRPL data set concatenation. This is to enable CICS to find the various programs that have been defined in the PPT.

You may also include a DD statement for ERXOUT extra partition data set. Once the startup JCL has been modified, restart your CICS system.

Build the ERXMAIN Control Block

Adapt the RPC server configuration in the Assembler program EMAINGEN (in EXP951.SRCE) to the customer environment. See *ERXMAIN Macro*.

Execute "Step 2: Build the ERXMAIN Control Block" of job EXPINSTA located in the EXX951.JOBS data set.

The name of the main control block in earlier versions was fixed as ERXMAIN. As of version 5.1.1, any meaningful name can be chosen. Using this name as input parameter *memory* for the *RPC Online Maintenance Facility* means that multiple CICS RPC Servers can be started and monitored in parallel. A fully linked control block named ERXMAIN is delivered in the load library EXP951.LOAD.

See also ***Verify the Installation of the EntireX RPC Server***.

TCP/IP-enable the CICS Region (Optional)

If you are using transport method TCP/IP, your CICS region must be enabled for TCP/IP. Refer to your CICS documentation for details.

Install the SVM File for a CICS RPC Server (Optional)

Execute "Step 7: Allocate and (CSD) define SVM file" of the EXPINSTA job located in the EXX951.JOBS data set. After updating the SVMFILE variable to a suitable name, the following steps are performed:

- allocation of the required VSAM cluster
- initialization of the cluster with the first SVM record fitting the CICS advanced channel container example DFHCON and the advanced CICS large buffer example DFHLBUF of the EXP951.DVCO data set. See *Client and Server Examples for z/OS CICS*.
- CICS CSD definition with the given name in the SVMFILE variable

See also *Handling SVM Files*.

Starting the EntireX RPC Server Automatically on CICS Startup (Optional)

To start the RPC server automatically on CICS startup

1 Insert a new PLT entry DFHPLT TYPE=ENTRY,PROGRAM=ERXSTART.

2 Rebuild the PLT for CICS startup.

Note: The EXP951.LD00 CICS load library contains a precompiled ERXSTART module with the default settings of the COBOL source member ERXSTART in EXP951.SRCE

Stopping the EntireX RPC Server Automatically on CICS Shutdown (Optional)

To stop the RPC server automatically on CICS shutdown

1 Insert a new PLT entry `DFHPLT TYPE=ENTRY,PROGRAM=ERXSTOP` into your CICS shutdown table.

2 Rebuild the PLT for CICS shutdown.

Note: The EXP951.LD00 CICS load library contains a precompiled `ERXSTOP` module with the default settings of the COBOL source member `ERXSTOP` in EXP951.SRCE

Installing Multiple EntireX RPC Servers in the same CICS (Optional)

To install a second RPC server in the same CICS

1 Copy the default CICS RPC Server transaction definition `ESRV` and give it a unique name, e.g. `ESR2`.

```
CEDA COPY TRANSACTION(ESRV) GROUP(ERX) TO(ERX2) AS(ESR2)
```

2 Copy the default CICS RPC Server *`ERXMAIN` Control Block* and give it a unique name, e.g. `ERXMAIN2`.

```
CEDA COPY PROGRAM(ERXMAIN) GROUP(ERX) TO(ERX2) AS(ERXMAIN2)
```

3 Add the new group `ERX2` to the CICS autoinstall list.

```
CEDA ADD GROUP(ERX2) LIST(listname) AFTER(groupname)
```

4 Build a new *`ERXMAIN` Control Block* and give it the name created above, e.g. `ERXMAIN2`.

As a minumum, set the *`ERXMAIN` Macro* parameter *`REPL`* in the `ERXMAIN` Control Block to the new RPC server transaction ID created above, e.g. `REPL=ESR2`.

The second CICS RPC server can now be started manually (see *Starting the RPC Server* under *RPC Online Maintenance Facility*), or automatically on CICS startup.

To start a second RPC server automatically on CICS startup

1 Copy the default CICS RPC Server autostart definition `ERXSTART` and give it a unique name, e.g. `ERXSTRT2`.

```
CEDA COPY PROGRAM(ERXSTART) GROUP(ERX) TO(ERX2) AS(ERXSTRT2)
```

2 "CEDA Install" the new autostart definition.

3 Modify the CICS RPC Server PLT startup routine `ERXSTART` from data set EXP951.SRCE.

- Update `RPC-TRANSID` with `ESR2`.
- Update `RPC-INPUT` with `MEM=ERXMAIN2`.

4 Compile and link the modified source and give it the name defined above, e.g. `ERXSTRT2`.

5 Insert a new PLT entry `DFHPLT TYPE=ENTRY,PROGRAM=ERXSTRT2`.

6 Rebuild the PLT for CICS startup.

To stop a second RPC server automatically on CICS shutdown

1 Copy the default CICS RPC server autostop definition `ERXSTOP` and give it a unique name, e.g. `ERXSTOP2`.

```
CEDA COPY PROGRAM(ERXSTOP) GROUP(ERX) TO(ERX2) AS(ERXSTOP2)
```

2 “CEDA Install” the new autostop definition.

3 Modify the CICS RPC Server PLT routine `ERXSTOP` from data set EXP951.SRCE.

- Update `RPC-TRANSID` with `ESR2`.
- Update `RPC-INPUT` with `MEM=ERXMAIN2`.

4 Compile and link the modified source and give it the name defined above, e.g. `ERXSTOP2`.

5 Insert a new PLT entry `DFHPLT TYPE=ENTRY,PROGRAM=ERXSTOP2` into your CICS shutdown table.

6 Rebuild the PLT for CICS shutdown.

Using SSL or TLS Connections with the CICS RPC Server (Optional)

See *Using SSL or TLS with the RPC Server*.

Installing EntireX RPC Servers under Batch

The EntireX RPC Server under z/OS (Batch, IMS) enables you to call COBOL or PL/I RPC services with standard IBM linkage convention as servers.

- Prepare Your Startup JCL
- Customize Your Server Configuration
- Using z/OS Privileged Services
- Install the SVM File for a Batch or IMS RPC Server (Optional)
- Using SSL or TLS Connections with the Batch RPC Server (Optional)

Prerequisites for all EntireX components are described centrally. See *z/OS Prerequisites* in the EntireX Release Notes.

Note: For Natural RPC servers, see *Setting Up a Natural RPC Environment* in your Natural documentation.

Prepare Your Startup JCL

The EntireX RPC Server for z/OS (Batch) can run as a started task. The installation medium contains the following sample JCL:

- EXPSRVB

To prepare your startup JCL

1. Modify the example job EXPSRVB of the EXX951.JOBS data set to suit your installation.
2. Modify the CONFIG DD statement to point to your server configuration file.
3. Concatenate your server application data set to the RPC server STEPLIB.
4. Add the EntireX RPC server JCL to your TASKLIB data set.

Customize Your Server Configuration

The EntireX RPC server is optimized for use in COBOL environments. Nevertheless, as a minimum the following parameters must be set according to your system environment:

- BrokerId
- Class
- ServerName
- Service

For more information see *Configuring the RPC Server* under *Administering the Batch RPC Server*.

Using z/OS Privileged Services

Some of the RPC server features such as impersonation require privileged z/OS access. Therefore, the RPC server should be started (initially) from an APF-authorized library. Consequently, all other load data sets concatenated to STEPLIB DD have to be APF-authorized as well, including customer's server data sets.

To cope with non-APF-authorized data sets, a server invocation module `EXXAUTH$` is provided. The module can be invoked (APF-authorized) from an APF data set (other than STEPLIB), which installs the authorized PC routines necessary before it invokes (unauthorized) the RPC server.

To install the server invocation module

1 Copy load module `EXXAUTH$` from EntireX load library *EXP951.LB00* to an APF-authorized load library that is linked to the system LNKLST concatenation and rename the copy to `EXXAUTH`.

Note: A different name is essential if the *EXP951.LB00* is part of the STEPLIB concatenation, because under z/OS search for the module is first done in the STEPLIB libraries prior to LNKLST libraries. Thus not changing the name would result in invoking the module from STEPLIB instead of invoking the module from LNKLST and the module could not work properly.

2 Change the RPC server startup JCL:

```
//BATRPCS  EXEC PGM=EXXAUTH, PARM='RPCSRVB CFG=DD:CONFIG'
```

Note: `EXXAUTH` expects the first parameter to be the name of the RPC server to be started. Subsequent parameters will be passed to the RPC server directly.

Install the SVM File for a Batch or IMS RPC Server (Optional)

Execute the `EXPSVMAL` job located in the EXX951.JOBS data set. After updating the `SVMFILE` variable to a suitable name, the following steps are performed:

- allocation of the required VSAM cluster
- initialization of the cluster with a dummy SVM record

Insert the new SVM file into your Batch RPC server's JCL under the DD name of `ERXSVM`. If this cluster is to be shared between more than one server it should be defined with the `RLS=NRI` attribute:

```
//ERXSVM   DD DISP=SHR,DSN=< batch.svm.cluster >
```

See *Handling SVM Files*.

Using SSL or TLS Connections with the Batch RPC Server (Optional)

See *Using SSL or TLS with the RPC Server* under *Administering the Batch RPC Server*.

Extract the EntireX RPC Examples from their Container Data Set

All example data sets are delivered in the condensed IBM `IEBCOPY` load format and can be found with their last level qualifier as a member name in the EXP951.EXPL data set. The CICS and batch examples are unloaded with separate jobs:

To unload the CICS members to their target data sets

- Select the desired programming language (COBOL or PL/I) and execute "Step 5: Extract the Examples" of job `EXPINSTA` located in the EXX951.JOBS data set.

To unload the Batch members to their target data sets

- Select the desired programming language (COBOL or PL/I) and execute job `EXPEXAMP` located in the EXX951.JOBS data set.

To unload the IMS members to their target data sets

- Execute job `EXPINSTI` located in the EXX951.JOBS data set.

6 Installing EntireX Security under z/OS

- Installing EntireX Security for Broker Kernel 42
- Setting up EntireX Security for Broker Stubs 45

Notes:

1. If you are using EntireX Security, and if your application(s) use ACI version 7 or below, you must install EntireX Security for Broker stubs. For ACI version 8 and above, see *Writing Applications using EntireX Security* in the ACI Programming documentation to ensure that your application(s) will perform as expected when using EntireX Security.
2. Before installing EntireX Security, make sure that all prerequisites for EntireX components have been met. See *z/OS Prerequisites* in the EntireX Release Notes.

Installing EntireX Security for Broker Kernel

This section describes the steps for installing EntireX Security for Broker kernel under z/OS. The installation procedure has the following steps:

- Modify Broker Attribute File
- Define RACF Resource Profiles
- Perform Client Authorization Checks (Optional)
- Enable Trusted User ID (Optional)
- Build Language-specific Messages (Optional)
- Start (Restart) Broker Kernel

Modify Broker Attribute File

To modify the Broker attribute file

1 Insert the following parameter in the section `DEFAULTS=BROKER` of the Broker attribute file:

```
SECURITY=YES
```

2 Modify the Security-specific attributes section of the Broker attribute file according to your requirements. These parameters are used to determine whether you will use SAF Security or LDAP-based authentication. See *Security-specific Attributes* (`DEFAULTS=SECURITY`) under *Broker Attributes* in the administration documentation. If you are using LDAP-based authentication, authorization checks are not available to you.

Note: Setting `SECURITY=YES` will load the provided LOAD module USRSEC from the EXX951.LOAD library. This module will perform privileged operations, such as execute the `RACROUTE`, requiring APF authorization.

Define RACF Resource Profiles

EntireX Security performs checks against user profiles and resource profiles represented in RACF, CA ACF2, and CA Top Secret. See *Resource Profiles in EntireX Security* in the EntireX Security documentation.

Perform Client Authorization Checks (Optional)

For services supporting Natural RPC or other applications that use RPC, you can perform authorization checks on the client by defining the "per service" attribute `CLIENT-RPC-AUTHORIZATION=YES` in the Broker attribute file. Setting this parameter to `YES` will cause the RPC library and program names to be appended to the profile associated with the authorization check. The resource profile would then appear as follows:

```
Class.server.service.rpc-library.rpc-program
```

If the total length of the resource profile exceeds 80 bytes, increase the parameter `MAX-SAF-PROF-LENGTH`.

This check applies only to the client and not the server. `CLIENT-RPC-AUTHORIZATION=YES` should not be set for any services which do not utilize RPC protocol.

Note: Natural Security performs its resource authorization checks as follows:
`<prefix-character>.rpc-library.rpc-program`
To allow conformity with Natural Security, the `CLIENT-RPC-AUTHORIZATION` parameter can optionally be defined with a prefix character as follows:
`CLIENT-RPC-AUTHORIZATION=(YES,<prefix-character>).`

Enable Trusted User ID (Optional)

If you use the trusted user ID option, set the parameter `TRUSTED-USERID=YES` in the `DEFAULTS=SECURITY` section of the attribute file.

- The trusted user ID feature automatically acquires the identity of the logged-on user or batch job. It must therefore only be used with TP monitors running under the control of RACF, CA ACF2 or CA Top Secret. Batch jobs must run under an identifiable user ID, as inherited by the job submitter, scheduler, or other means.
- Applications using the trusted user ID feature must execute under z/OS and on the same machine, or another z/OS machine connected to Broker through Entire Net-Work. Communication is through the Adabas SVC mechanism.
- Applications must not assign a password to the ACI control block if they intend to use trusted user ID. This applies to all applications, including EntireX RPC Server and EntireX Broker Services (Attach Services). If the application cannot avoid supplying a password, it is permissible to assign a password value of `NOPASSWORD`, as required by the Broker Services SDL file.

- EntireX Security trusted user ID functionality is relevant only for determining the z/OS user ID associated with applications executing on z/OS which communicate with EntireX Broker or Broker Services, which are also executing on z/OS via the Adabas SVC mechanism. It cannot be used in configurations which include application components executing on separate, non-z/OS computers that communicate with EntireX Broker or Broker Services through Entire Net-Work. Such configurations invalidate the usage of trusted user ID.
- The SVCSAF module is supplied with EntireX. If your Adabas version is lower than 8.2, the resulting Adabas SVC must be linked into an APF authorized library. Since Adabas 8.2, the SAF component is linked to ADASVC by default. Linkage example:

```
//*-------------------------------------------------------
//* CREATE A NEW ADASVC MODULE THAT INCLUDES SVCSAF MODULE
//*-------------------------------------------------------
//LNKSVC EXEC PGM=IEWL,PARM='XREF,LIST,LET,NCAL,RENT,REUS'
//SYSPRINT DD SYSOUT=*
//SYSUT1  DD SPACE=(CYL,(1,1)),UNIT=VIO
//WALLIB  DD DISP=SHR,DSN=WAL825.LOAD      ADASVC and SVCSAF
//SYSLMOD DD DISP=SHR,DSN=WAL825.NEW.LOAD  NEW ADASVC OUTPUT
//SYSLIN  DD *
MODE AMODE(31),RMODE(24)
INCLUDE WALLIB(ADASVC)
SETCODE AC(1)
INCLUDE WALLIB(SVCSAF)
NAME ADASVC(R)
/*
//*
```

- Implementing the SAF trusted user ID option in EntireX Security under CICS TS version 1.2 and above requires the installation of the Adabas task related user exit (ADATRUE) and setting either the ADAGSET or LGBLSET (depending upon the Adabas version) parameter `SAF=YES`. See *Installing Adabas with TP Monitors* in the Adabas installation documentation for complete details on installing ADATRUE. Samples of the ADAGSET and LGBLSET parameter modules can be found in the library WAL825.SRCE.

For additional supporting information, see the *Installation Procedure* section of the *Adabas Installation Manual*.

Build Language-specific Messages (Optional)

To build language-specific messages

1 Copy the template message module EXX951.SRCE(NA2MSG0) to another member - for example, EXX951.SRCE(NA2MSG9) - and then modify the message texts to suit your own language requirements.

Note: NA2MSG0, NA2MSG1, and NA2MSG2 are reserved names.

2 Assemble and link your modified source module using the sample JCL EXX951.SRCE(SAGJ106), ensuring that you create a unique load module in the EXX951.LOAD library.

3 Modify the `ERRTXT-MODULE` parameter in the `DEFAULTS=SECURITY` section of the attribute file to reflect the name of your unique load module.

Start (Restart) Broker Kernel

The Broker must be restarted to pick up changes to the Broker attribute file and to initialize Broker kernel under z/OS to perform security checks.

Basic installation of EntireX Security for Broker kernel is now complete.

Setting up EntireX Security for Broker Stubs

This section describes the steps for installing EntireX Security for Broker stub under z/OS. The installation consists of the following steps:

- Assemble the SAFCFG Configuration Module
- Link the Security Components
- Rename SECUEXIT

Notes:

1. If you are running your application(s) at ACI version 7 or below, the following steps are required to install EntireX Security for the Broker stubs in all environments where applications execute either as clients or servers. See *List of Components per Platform* under *Platform Coverage* in the EntireX Release Notes to see where EntireX Security for broker stubs is supported.
2. The mainframe stubs now employ high performance direct cross memory to send and receive data from buffers in the application's working storage. This is utilized for sending/receiving more than 32 KB of data. Therefore, when encryption is active, the application programmer must not rely on the contents of the `SEND` buffer after issuing the `SEND` command, since the

contents of the SEND buffer will be encrypted when sending more than 32 KB of data. We recommend you code all applications so that you do not rely on the contents of the SEND buffer after calling Broker. This will be required in the future for all SEND commands regardless of whether the data exceeds 32 KB. Therefore, the application's SEND buffer must not be in read-only memory, where encryption is activated.

These steps are not required if you are running your application(s) at ACI version 8 or above.

Assemble the SAFCFG Configuration Module

The SAFCFG configuration module is required for applications running on z/OS using ACI version 7 or below.

To assemble the SAFCFG configuration module

- Run job WAL&*vrs*..JOBS(SAFI010), which assembles and links SAFCFG (load module).

  Note: This module comes with preconfigured defaults. See source module WAL&*vrs*..SRCE(SAFCFG). If encryption is required, set the macro assembly parameter as follows: BKPRIV=1.

Link the Security Components

For applications running on z/OS using ACI 7 or below, the Broker stub security component must be linked with the following stubs: BROKER, CICSETB, NATETB23, COMETB, MPPETB.

To link the Broker stub security component

- Relink all applications that contain ACI stub modules BROKER, CICSETB, NATETB23, COMETB, or MPPETB to include the following modules:

  - NA2PETS Broker security stub logic module
  - SAFCFG System parameter module

  See *Administration of Broker Stubs under z/OS* in the z/OS administration documentation.

  Location of sample INCLUDE statements: EXX&*vrs*.JOBS(EXXJ109).

  Note: These components are needed for backward compatibility if your applications issue any commands using ACI version 7 or below. Applications using ACI version 8 or above do not require these additional components in the stubs.
  For ACI version 7 or below, these components must be added to the stub environment utilized by the application. Failure to link these components along with the stub when

using ACI version 1 though 7 can result in message "SEFM225 MESSAGE FROM BACK LEVEL STUB" being issued by Broker kernel.

Rename SECUEXIT

SECUEXIT must be made available for applications running on z/OS using ACI version 7 or below.

To make SECUEXIT available

1 Rename SECUEXIO to SECUEXIT in library EXX&*vrs*.LOAD so that it is available to applications running the IBM C stub.

2 Ensure that SECUEXIT is available in EXX&*vrs*.LOAD for all applications.

Notes:

1. These steps are needed for backward compatibility if your applications issue any commands using ACI version 7 or below. Applications using ACI version 8 or above do not require these additional components in the stubs.
2. For ACI version 7 or below, these components must be added to the stub environment utilized by the application.

Installation of EntireX Security for Broker stubs is now complete. Now you can install the security components for the Broker stubs on the remaining operating systems where your application components are located.

See also *Setting up EntireX Security for Broker Stubs* in the UNIX|Windows installation section of the documentation.

7 Installing EntireX Java Components under z/OS UNIX

- Scope 50
- Installation Steps 50
- Configuring and Administering the Java Components 51

This chapter describes how to install EntireX Java components under z/OS UNIX.

Scope

The following EntireX Java components can be used under z/OS UNIX:

- Java RPC Server/Client
- XML/SOAP RPC Server/Client
- WebSphere MQ RPC Server
- WebSphere MQ Listener
- RPC-ACI Bridge
- XML/SOAP Listener
- CICS ECI RPC Server
- IMS Connect RPC Server

Installation Steps

To install EntireX Java components under z/OS UNIX

1 Unload the delivered installation medium to your DASD device. See *Copying the Contents of the Installation Medium to Disk*.

2 On your UNIX system, create a directory for the installation. For example:

```
mkdir SoftwareAG; cd SoftwareAG;
```

3 Copy the file *hlq*.EXX*vrs*.TAR to the created directory, where *hlq* is your high-level qualifier and *vrs* is the product version, release and service pack:

```
tso "oput 'hlq.EXXvrs.TAR' 'exxvrs.tar' binary"
```

4 Unpack the TAR file:

```
tar xvf exxvrs.tar
```

Now you have created a folder structure with subdirectories EntireX and WS-Stack which contain all necessary files in several subdirectories. In the `EntireX/bin` directory you have an environment file `exx_zos_env.sh`, which you can source to set your environment. Before sourcing the environment, edit this file and set the environment variable `EXXDIR` to your EntireX directory. Then you can enter the following command (note the space after the initial period):

```
. ./exx_zos_env.sh
```

Check your Java installation and edit the file `exxjenv.sh` in the `EntireX/bin` directory. You need to set the `JAVA_HOME` environment variable to your corresponding Java installation.

Bourne shell scripts are provided in the `EntireX/bin` directory to start the different RPC servers.

To use the XML/SOAP RPC Server, configure the path to the `example.xmm` file. Edit the file `EntireX/config/entirex.xmlrpcserver.configuration.xml` and adapt the pathname in the `exx-xmm` tag. This file is already in EBCDIC format and the encoding has been set to CP037. If you need a different encoding, change this setting accordingly.

Configuring and Administering the Java Components

The following links provide more information on configuring and administering the EntireX Java components under UNIX.

EntireX Java Component	More Information
Java RPC Server	*Administering the EntireX Java RPC Server* in the UNIX and Windows administration documentation
WebSphere MQ RPC Server and WebSphere MQ Listener	*EntireX WebSphere MQ RPC Server*
RPC-ACI Bridge	*EntireX RPC-ACI Bridge*
XML/SOAP RPC Server	*Administering the EntireX XML/SOAP RPC Server* in the UNIX and Windows administration documentation
XML/SOAP Listener	*Configuring the XML/SOAP Listener* in the UNIX and Windows administration documentation
CICS ECI RPC Server	*CICS ECI RPC Server*
IMS Connect RPC Server	*IMS Connect RPC Server*

The following table shows the provided start script and configuration file for each RPC server.

RPC Server	Script in EntireX/bin	Config File in EntireX/config
Java RPC Server	`jrpcserver.bsh`	`entirex.javarpcserver.properties`
XML/SOAP RPC Server	`jxmlrpcserver.bsh`	`entirex.xmlrpcserver.properties`
WebSphere MQ RPC Server	`wmqbridge.bsh`	`entirex.wmqbridge.properties`
WebSphere MQ Listener	`wmqlistener.bsh`	`entirex.wmqbridgelistener.properties`
RPC-ACI bridge	`jrpcacibridge.bsh`	`entirex.rpcacibridge.properties`
CICS ECI RPC Server	`cicseciserver.bsh`	`entirex.cicseci.properties`
IMS Connect RPC Server	`imsconnectserver.bsh`	`entirex.imsconnect.properties`

For the XML/SOAP Listener you need to set up a Web server and install Software AG Common Web Services Stack to this Web server. See *Deployment of Web Services Stack Web Application Runtime* in section *Administration* of the *Web Services Stack* documentation. After successful installation, copy or upload `<inst_dir>/EntireX/classes/entirex.jar` to the WEB-INF/lib folder in the Web Services Stack application folder.

8 Verifying the z/OS Installation

- Testing the EntireX Broker Interface Installation 54
- Verify the Installation of the EntireX RPC Server 54

Once you have installed the EntireX Broker and EntireX Developer's Kit, you can verify that the installation was successful.

Testing the EntireX Broker Interface Installation

To test the EntireX Broker Interface installation

1 Start EntireX Broker. See *Setting up Broker Instances*.

2 Run the example programs:

- *BCOS* is the example server program
- *BCOC* is the example client program

The execution JCL for the example programs is located in the EXX951.JOBS library in members BCOSJCL and BCOCJCL. Modify the JCL for your installation, and submit job BCOSJCL first, followed by job BCOCJCL.

A single message will be sent by program BCOC and then received by program BCOS. Both jobs should return a zero condition code. The output from each job will be written to SYSPRINT.

Verify the Installation of the EntireX RPC Server

The installation can be verified by running the EXAMPLE program provided with EntireX Developer's Kit on the PC, located in the EntireX directory under Examples/Broker RPC/Client/.

This verification requires that the Broker is up and running and the RPC server has been started. See *Setting up Broker Instances* and, depending on the RPC server,

- *Starting the RPC Server*
- *Starting the RPC Server* under *Administering the Batch RPC Server*
- *Starting the EntireX RPC Server (IMS)* under *Administering the EntireX RPC Server under z/OS IMS*

For example, if you are using the CClient, enter the command

```
Client ETB001 RPC SRV1 CALLNAT
```

where ETB001 is the Broker ID of the server

RPC, SRV1, and CALLNAT are the Broker CLASS, SERVER NAME, and SERVICE being offered

the selection panel is displayed:

```
CMD        Function
-----------------------------------------
C          RPC CALC function
S          RPC SQUARE function
I          Ping the running RPC server
T          Terminate the running RPC server
.          Exit
-----------------------------------------
Please select:
```

Choose option C (RPC CALC function), where you should receive the response:

```
Calling CALC: 12345 + 67890 = 80235.
```

9 Installing Adabas Components for EntireX under z/OS

- Initializing the Adabas Communication Environment 58
- SVC Integrity Validation 68
- Requirements for Cross-Memory Services 69
- Applying Zaps 70

Initializing the Adabas Communication Environment

This section describes the installation of the Adabas router (ADASVC). The router uses cross-memory services for communication between the Adabas nucleus and the Adabas users.

The Adabas z/OS cross-memory communications service comprises two modules:

- the Adabas router (ADASVC); and
- the Adabas subsystem initialization routine (ADASIR).

ADASIR, executed either during IPL or by the Adabas SVC installation program (ADASIP), initializes the router's operating environment, particularly the ID table.

ADASVC installation can be either temporary or permanent:

- The Adabas SVC can be installed temporarily by executing ADASIP. The SVC is then available only until the next IPL.

  Note: Once installed, the Adabas SVC can be reinstalled temporarily using the ADASIP REPLACE option. However, no Adabas nucleus can be active during this procedure.

  Note: It is necessary to cycle CICS after executing ADASIP to initialize the SVC.

- The Adabas SVC is installed permanently using regular operating systems procedures. The SVC then requires an IPL to become active.

Typically, the Adabas SVC is first installed temporarily using ADASIP. This makes Adabas available immediately without the need to wait for an IPL. Meanwhile, preparations are usually made for permanent installation at the next IPL.

- SVC Compatibility Issues Between Adabas Releases
- APF Authorization Requirement
- Allocating an SVC Table Entry
- Subsystem Name Requirements
- Page-Fixing the Adabas SVC
- Initializing the Adabas SVC
- Router Installation Overview
- Using ADASIP for Temporary Installations
- Using ADASIR
- Relinking the SVC for Temporary Installation

- Relinking the SVC for Permanent Installation

SVC Compatibility Issues Between Adabas Releases

Adabas 8 includes a new Adabas SVC. This SVC is fully backward compatible. In other words, you can use the new Adabas 8 SVC with Adabas 7 (or earlier) databases.

However, you *cannot* use the Adabas SVC from previous Adabas releases with Adabas 8 databases. If you attempt to do this, the Adabas 8 database will not initialize successfully.

The Adabas 8 SVC includes performance improvements and improved error recovery routines. Note that the new SVC uses more efficient operating system interfaces, in particular when posting the user at command completion. This shifts work from SRB-mode routines to TCB-mode routines and also between the user's program and the Adabas nucleus. Take this into account when analyzing Adabas 8 SVC performance. With the new SVC, SRB-mode overhead is largely eliminated and TCB-mode overhead is somewhat increased, but the net result is an overall improvement in SVC performance.

APF Authorization Requirement

The Adabas 8 SVC requires that the Adabas nucleus, as well as other MPM servers (such as Entire Net-Work and the Natural Global Buffer Pool), be APF-authorized. This APF authorization prevents unauthorized use of the ADASVC 0-call. Software AG recommends strongly that you run APF-authorized because of the security risks you can incur if you do not. However, upon request, Software AG does have a zap (AY811031) you can apply that eliminates this requirement.

Note: Some add-on products require APF authorization to use restricted z/OS services. APF authorization is still required in these cases.

Allocating an SVC Table Entry

Regardless of the installation procedure selected, an available SVC table entry must be allocated to the Adabas router (ADASVC). SVC table entries are defined in the member IEASVCxx of SYS1.PARMLIB.

The SVC table entry in the operating system for an ADASVC must contain the following information:

Offset	Label	Description
0	SVCEP	SVC entry point address.
4	SVCATTR1	Must indicate type 2 SVC (flag bit SVCTP2 set-X'80') or type 3 or 4 SVC (flag bits SVCTP34 set-X'C0'): ADASIR changes a type 1, 5, or 6 SVC to type 2. May indicate that APF-authorization is needed for this SVC (flag bit SVCAPF set—X'08'): if set, all targets and users must be APF-authorized.

Offset	Label	Description
6	SVCLOCKS	Must contain all zeros. ADASIR sets SVCLOCKS to zeros.

Subsystem Name Requirements

The subsystem name contained in the four-character field SUBSYS at ADASVC offset X'28' (the default is "ADAB") must be the same as that specified in the IEFSSN*xx* member of SYS1.PARMLIB. If the name is not the same, ADASIR ends with an ADAS12 message and condition code 2, and Adabas is not usable.

Page-Fixing the Adabas SVC

If the Adabas SVC is to reside in the fixed LPA, add an entry to an IEAFIX*xx* member of SYS1.PARMLIB.

Initializing the Adabas SVC

The Adabas SVC should be initialized with ADASIP/ADASIR in order to guarantee full functioning of all Adabas nuclei.

Router Installation Overview

- Temporary Router Installation (SMA Job Number I011)
- Permanent Router Installation (SMA Job Number I010)

Temporary Router Installation (SMA Job Number I011)

Once you have restored the Adabas installation medium, use a local editor to customize the job JCLLINK (used to link ADASIR, ADASIP, and ADASVC) as follows:

To perform temporary router installation

1 Link ADASIP into an APF-authorized library as an authorized module.

2 Link ADASIR and ADASVC into APF-authorized libraries:

- Place ADASVC in an APF-authorized library in order to run ADASIP.
- Place ADASIR in an APF-authorized library concatenated to SYS1.LINKLIB defined in source member LNKLST*xx* located in SYS1.PARMLIB.

3 Execute ADASIP to install the SVC.

Customize and run the job ADASIP to dynamically add the Adabas SVC without an IPL.

Permanent Router Installation (SMA Job Number I010)

To perform permanent router installation

1 Link the Adabas SVC (ADASVC) which has been renamed according to the SVC routine renaming rules (for example, type 3 SVCs must have names of IGC00*nnn*, where *nnn* is a signed decimal SVC number) into SYS1.LPALIB as a permanent step for ADASIR.

2 Link ADASIR into SYS1.LINKLIB or into an APF-authorized library concatenated to SYS1.LINKLIB with the LNKLST*xx* member of SYS1.PARMLIB.

Note: ADASIR is not reentrant, and therefore should not be linked into SYS1.LPALIB.

3 Customize and run the job JCLUPDT to add a new entry with the correct format.

4 IPL z/OS with the CLPA option to install and initialize the Adabas communication environment.

Using ADASIP for Temporary Installations

- ADASIP Functions
- ADASIP Parameters
- Executing ADASIP

ADASIP Functions

ADASIP performs the following functions:

- acquires memory in the specified CSA subpool for the Adabas SVC and a subsystem communication vector table (SSCT)
- loads the Adabas SVC into the acquired CSA space
- modifies the SVC table entry as required by the Adabas SVC
- optionally deletes an SSCT for the same subsystem name from the SSCT chain
- adds the new SSCT to the SSCT chain
- invokes the ADASIR program
- releases CSA acquired by a previously installed SVC

If any error is detected, ADASIP backs out all completed activities and terminates operation with a user abend specifying the error.

When reinstalling an instance of ADASVC using an SVC number that is currently being used by ADASVC, the subsystem name must be the same as the one currently being used. This helps avoid a configuration that may not function correctly. For more information, read ***SVC Integrity Validation***, elsewhere in this guide.

A Version 8 ADASIP/ADASIR can be used to install a Version 7 Adabas SVC, but the Version 8 SVC requires the Version 8 ADASIP/ADASIR. Software AG recommends using Version 8 to install all supported releases of the SVC. If an earlier version of ADASIP/ADASIR is used to replace an SVC installed with a later version, some areas of common storage may not be released.

The following JCL links ADASIP, located in ADABAS.V*vrs*.LOAD, into an APF-authorized library as an authorized module:

```
//LNKSIP EXEC PGM=IEWL
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//ADALIB DD DSN=ADABAS.V vrs .LOAD,DISP=SHR
//SYSLMOD DD DSN=apflibname,DISP=SHR
//SYSLIN DD *
INCLUDE ADALIB(ADASIP)
SETCODE AC(1)
NAME ADASIP(R)
```

ADASIP Parameters

ADASIP parameters have the following syntax:

```
CONSNAME=c,IDTSPL=i,LEAVE=l,NRIDTES=n,REPLACE=r,SUBSYS=su,
SVCNR=svcn,SVCSPL=svcs
```

-where

c	is the console name to which operator messages are written. If omitted, messages are issued using ROUTCDE=2, master Console Information.
i	is the ID table subpool: see the ADASIR `IDTSPL` parameter for details.
l	indicates whether ADASIR should display message ADAS11 or ADAS12 on the operator console: see the ADASIR `LEAVE` parameter for details.
n	is the number of ID table entries: see the ADASIR `NRIDTES` parameter for details.
r	indicates whether or not an existing SSCT for the same subsystem name is to be replaced. Y for yes or N for no (N is the default). Use this option to replace any type of Adabas SVC (for example, when installing a new SVC version).
su	is the subsystem name. This parameter is required. . Each instance of the Adabas SVC must have a unique subsystem name.
svcn	is the Adabas SVC number: see the ADASIR `SVCNR` parameter for details.
svcs	is the Adabas SVC and SSCT subpool: 228 for fixed CSA or 241 for pageable CSA (default: 241).

The following are valid ADASIP parameter abbreviations:

Parameter	Abbreviation
CONSNAME=	C=
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
REPLACE=	R=
SUBSYS=	SU=
SVCNR=	SVCN=
SVCSPL=	SVCS=

All parameters are optional except SUBSYS and SVCNR. If specified, the parameters IDTSPL,LEAVE, NRIDTES, SUBSYS, and SVCNR are passed to ADASIR without being verified.

Executing ADASIP

JCL similar to the following should be used to execute ADASIP:

```
// EXEC PGM=ADASIP,PARM=parameters
//STEPLIB DD ...
//SVCLIB DD ...
//SIRLIB DD ...
```

The data set defined by the STEPLIB DD statement must be an APF-authorized library containing the APF-authorized program ADASIP. Since ADASIP is neither reentrant nor refreshable, the data set cannot be SYS1.LPALIB.

The data set defined by the SVCLIB DD statement must be an APF-authorized library containing the Adabas SVC with either the name or alias ADASVC.

The data set defined by the SIRLIB DD statement must contain the ADASIR program. Since ADASIR is neither reentrant nor refreshable, the data set may not be SYS1.LPALIB.

ADASIP terminates with a U0481 abend if the parameter input is incorrectly specified.

The IBM job control convention for continuing the PARM parameter is:

```
// EXEC PGM=ADASIP,PARM=('parameters ....', X
// 'parameters')
```

-where X in column 72 is a continuation character. The following restrictions also apply to JCL statements:

- a comma is required after the end-quote on a line that is to be continued
- a non-blank continuation character is required in column 72 of each line that is to be continued, and the continuation line must start within columns 4-16

- a comma is not permitted between the last parameter and the end-quote on the line to be continued because JCL automatically inserts a comma between parameters when concatenating continuation strings:
  - ```
    // ...PARM=('CONSID=3', X
    // 'SUBSYS=ADAB', X
    // 'SVCNR=249')
    ```

    -results in an equivalent line of

    ```
    CONSID=3,SUBSYS=ADAB,SVCNR=249
    ```

Using ADASIR

- ADASIR Functions
- Relinking ADASIR
- ADASIR Parameters
- Executing ADASIR

ADASIR Functions

The ADASIR program is invoked

- by the ADASIP program to install the Adabas SVC temporarily, or
- by z/OS to install the Adabas SVC permanently.

ADASIR receives control during either master scheduler initialization or ADASIP execution. The operator is prompted for any value that has been incorrectly zapped or assembled (refer to the *Adabas Messages and Codes* for specific message descriptions). If an error is found during the processing of parameters specified in the IEFSSN*xx* member or passed by ADASIP, the operator is prompted for all of the values.

If the SVC table entry is incorrect, ADASIR prompts the operator for permission to change the entry (if `SVCTAB=P`, the default, is specified). If any errors are detected, they must be corrected and either another IPL must be done or ADASIP must be rerun before the Adabas SVC can be used.

Relinking ADASIR

The ADASIR module must be linked into an APF-authorized library.

The following JCL links ADASIR, located in ADABAS.V*vrs*.LOAD, into SYS1.LINKLIB:
    ```

```
//LNKSIR EXEC PGM=IEWL
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=(CYL,(1,1)),UNIT=3350
//ADALIB DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
INCLUDE ADALIB(ADASIR)
NAME ADASIR(R)
```

ADASIR Parameters

ADASIR parameters have the following syntax:

```
IDTSPL=i,LEAVE=l,NRIDTES=n,SVCNR=svcn,SVCTAB=svct
```

Variable	Description
i	The ID table subpool: 228 for fixed CSA or 241 (the default) for pageable CSA.
l	Indicates whether message ADAS11 or ADAS12 is to be displayed on the operator console: Y for yes or N (the default) for no.
n	The ID table entry count, which can range from 1 to a maximum specified at offset X'146' in the CSECT IEAVESVT of the z/OS nucleus (see section *Requirements for Cross-Memory Services*).
svcn	The Adabas SVC number (200-255).
svct	Indicates whether or not the operator should be prompted for permission to update the SVC table entry. Enter P (the default) to receive a prompt, or N for no prompt. P is recommended if a possibility exists that the SVC table entry will not be what ADASIR expects.

The following are valid abbreviations for ADASIR parameters:

Parameter	Abbreviation
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
SVCNR=	SVCN=
SVCSPL=	SVCS=

Executing ADASIR

Note: The ADASIR module must be linked into an APF-authorized library.

To prepare for permanent SVC installation, an entry must be made in either a new or existing member having the name IEFSSN*xx* in SYS1.PARMLIB. This entry is an 80-character record with the following format:

```
SUBSYS SUBNAME(cccc) CONSNAME(consname) INITRTN(ADASIR) ↩
INITPARM('parameters') comments
```

-where

cccc	The 1- to 4-character subsystem name. This name and the name specified in the Adabas SVC at offset X'28' must be the same. The name provided in the SVC is ADAB; any other name must first be zapped into the SVC before being specified for *cccc*.
consname	The name of the console to which ADASIR will direct any messages. If omitted messages will be issued with ROUTCDE=2, Master Console Information.
'parameters'	ADASIR parameters. If there is more than one parameter, values must be enclosed in single quotation marks and a comma placed between the parameters.
comments	Comments are optional and must be preceded by at least one space.

If the subsystem name does not match, ADASIR abends with an ADAS12 message and condition code 2; the Adabas z/OS communication environment is not initialized. Re-IPL z/OS, specifying `SSN=`*xx* if necessary. If this is the first IPL with a type 3 or 4 Adabas SVC, specify CLPA as one of the `SET` parameters.

If an error is encountered while processing any of the parameters obtained from the IEFSSN*xx* member or passed from ADASIP (message ADAS05), the operator is prompted to reenter all of the parameters. If the SVC table entry is not correct (message ADAS09) then, depending on the value of the `SVCTAB` parameter, either the operator is prompted (message ADAS10) for permission to change the `SVCTAB` parameter, or it is simply changed (message ADAS15).

A Version 8 ADASIR can be used to install a Version 7 Adabas SVC, but the Version 8 SVC requires the Version 8 ADASIR. Software AG recommends using Version 8 to install all supported releases of the SVC.

Relinking the SVC for Temporary Installation

Link the Adabas SVC with the name or alias ADASVC into an APF-authorized library. ADASVC must be linked with `AMODE=31` and `RMODE=24`, the default.

The following example shows how to link the SVC:

```
// (job card)
//LKED EXEC PGM=IEWL,
// PARM='XREF,LIST,NCAL,LET,MAP,RENT,REFR,REUS'
//SYSPRINT DD SYSOUT=X
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR <--Target loadlib
//ADALIB DD DSN=user.loadlib,DISP=SHR <--ADASVC loadlib
//SYSLIN DD *
MODE AMODE(31),RMODE(24)
INCLUDE ADALIB(ADASVC)
NAME ADASVC(R)
/*
```

Note: If the SVC is linked with a name other than ADASVC when preparing to upgrade to a permanent installation, the SVC must have an alias of ADASVC. When dynamically loading the Adabas SVC, ADASIP searches for the module ADASVC in the library specified by the SVCLIB DD statement.

Relinking the SVC for Permanent Installation

Software AG recommends using a type 3 or 4 SVC for the Adabas SVC.

SVC types 1 and 6 are not supported.

- Type 2 SVC
- Type 3 or 4 SVC

Type 2 SVC

If the Adabas SVC is to be type 2, link it into SYS1.NUCLEUS as the system nucleus IEANUC0*x*.

This nucleus must contain an SVC table entry for an enabled type 2 SVC, which must be defined during SYSGEN.

Then include linkage editor control statements similar to the following with those needed to link a nucleus:

```
CHANGE ADASVC(IGCnnn) ---> nnn is the SVC number in decimal
INCLUDE ADALIB(ADASVC) ---> ADALIB contains the Adabas SVC
```

Type 3 or 4 SVC

To install the Adabas SVC as type 3 or 4, link the Adabas SVC with the appropriate name into SYS1.LPALIB. ADASVC must be linked with `AMODE=31` and `RMODE=24` (the default).

The following example shows how to relink the SVC:

```
// (job card)
//LKED EXEC PGM=IEWL,
// PARM='XREF,LIST,NCAL,LET,MAP,RENT,REFR,REUS'
//SYSPRINT DD SYSOUT=X
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DSN=SYS1.LPALIB,DISP=SHR <--Target Loadlib
//ADALIB DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR <--ADASVC loadlib
//SYSLIN DD *
MODE AMODE(31),RMODE(24)
CHANGE ADASVC(IGCOOnnp) <- where nn are the first two digits of the SVC in decimal and
INCLUDE ADALIB(ADASVC) p is the character corresponding to the x'Cn'
NAME IGCOOnnp(R) ----> (n is the last digit of the SVC number in decimal)
*
/*
```

SVC Integrity Validation

In the past, the presence of multiple SVCs with the same subsystem ID has resulted in a single ID table being used by different SVCs. This has caused problems, some of them serious (abnormal nucleus termination or corruption of the database).

To eliminate this danger, the Version 8 SVC checks to ensure that the SVC accessing the ID table is the same as the one that was used by ADASIP/ADASIR to initialize the table. If the SVCs are not the same, an abend 650 occurs.

Abend 650 occurs when an incorrect SVC number is specified in the ADARUN parameters for a nucleus. It can occur during Adabas initialization, during the first Adabas call from a user program, or when the ID table is queried by another Software AG server such as Entire Net-Work.

ADASIP has been enhanced to prevent this from arising. If you attempt to install an instance of ADASVC using an SVC number that is presently associated with another subsystem name, ADASIP will terminate with abend 435.

Requirements for Cross-Memory Services

Due to the implementation of cross-memory services in z/OS, the following points should be noted when running an Adabas nucleus in MULTI mode:

- a maximum of one step of a job can establish the cross-memory environment. This means that a job can include at most one step that is a target (for example, an Adabas nucleus).
- cross-memory accesses may not be made to a swapped-out address space. Therefore, the address space of an Adabas nucleus is set to "nonswappable" for the duration of the nucleus session. This can increase the installation's real storage requirements. This behavior is documented in the IBM manual *Extended Addressability Guide*, chapter Synchronous Cross-Memory Communication.
- when a nucleus with an active cross-memory environment terminates either normally or abnormally, the entire address space including any initiator is also terminated.

The ASID representing this address space is not reassigned until the next IPL. Therefore, you should choose a sufficiently high value for the `MAXUSERS` parameter in the active IEASYS*xx* member of SYS1.PARMLIB or - if your system supports it - the `RSVNONR` parameter in the same member can be adjusted accordingly. Also, the Adabas nucleus should not be stopped and started without good reason.

This is described in the manuals referred to in the topics Recovery Considerations and Resource Management. Additional information can be found in IBM APARs OZ61154, OZ61741, and OZ67637.

To make its services available to all address spaces in the system, the Adabas nucleus must obtain a system linkage index (LX) from z/OS. The LX is a reserved slot in the linkage space of all address spaces, and permits system-wide linkage to all address spaces.

If your configuration is using z/OS 1.6 or later and your hardware supports the Address Space and LX Reuse Facility (ALRF), the version 8 ADASVC will make use of reusable system LXs. Otherwise, a non-reusable system LX will be used as in previous releases. Reusable system LXs resolve the constraints imposed by non-reusable LXs. The remainder of this section discusses these constraints.

The number of non-reusable LXs set aside by z/OS for system use is rather small (usually 165 out of a possible 2048).

Because of the way z/OS use cross-memory services, non-reusable system LXs obtained by Adabas cannot be returned to z/OS until the next IPL. However, the system that owns the LXs can reuse them, even for a different address space. Adabas makes use of this feature by identifying used LXs in a table in CSA, where they are available to future nuclei.

The number of non-reusable system LXs can be specified in the member IEASYSxx contained in SYS1.PARMLIB, using the NSYSLX parameter. If you change this value, you must perform an IPL to make the change effective.

To determine an appropriate NSYSLX value, consider the following points:

- some LXs are probably already being used by other system functions. Therefore, the chances of creating an LX shortage for other users is small.
- Adabas requires one system LX for each Adabas nucleus (or any other target) that will be active concurrently. A value of decimal 64 would allow concurrent execution of up to 64 Adabas nuclei or other targets with little chance of restricting other components using LXs.
- Entire Net-Work Version 5 uses only one LX and one ID table entry, regardless of how many remote databases it must represent. This is unlike the pseudo-MPM concept of earlier Entire Net-Work versions.
- whenever ADASIP is executed with the REPLACE option, all LXs saved in the current ID table are lost until the next IPL.

Likewise, if a session ends either normally with the FORCE operator command or abnormally during ESTAE processing (for example, by an S222 operator cancel or by a S722 spool limit exceeded abend during a snap dump), the LX also cannot be recovered until the next IPL.

Any commands sent to these targets receive an S0D6 abend. Any attempt to restart the nucleus results in an ADAM98 message DUP ID (LOCAL), followed by an abend. To resolve both of these problems, restart the nucleus with the ADARUN FORCE=YES and IGNDIB=YES parameters.

The first target that tries to obtain a system LX when none is available ends with an S053 abend code and reason code 0112. No additional targets can be started until the next IPL.

Applying Zaps

Use the z/OS AMASPZAP utility to apply zaps in the respective operating system; this method verifies (VER) and replaces (REP) data. The following sample JCL executes AMASPZAP:

```
//ADAZAP JOB
//STEP1 EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=X
//SYSLIB DD DSN=ADABAS.V vrs .LOAD,DISP=SHR
//SYSIN DD *
(zap control statements)
/*
//
```

-where the following are examples of zap control statements:

```
NAME membername csectname
VER displacement data
REP displacement data
IDRDATA (up to eight bytes of user data)
* (comment)
```

Note: In VER and REP statements, spaces must be used to separate command, displacement, and data. Commas are acceptable data separators; however, commas with spaces or spaces alone are not, and may cause errors.

10 Installing Adabas with TP Monitors

- Preparing Adabas Link Routines for IBM Platforms 74
- Installing Adabas with IMS/TM under Adabas 8 75
- General Considerations for Installing Adabas with CICS 77
- Installing Adabas with CICS under Adabas 8 79
- Installing Adabas with Com-plete under Adabas 8 82
- General Considerations for Installing Adabas with Batch/TSO 84
- Installing Adabas with Batch/TSO under Adabas 8 86
- Modifying Source Member Defaults (LGBLSET Macro) in Version 8 88

This chapter provides information needed to install Adabas in batch mode and with its teleprocessing (TP) monitors.

Preparing Adabas Link Routines for IBM Platforms

This section describes the preparation of Adabas link routines for TP monitors for IBM platforms. The source modules for Adabas 8 link routines are not provided in the Adabas 8 base source library. The Adabas 8 link routines can only be tailored via zap or using a link globals table.

- Addressing Mode Assembly Directives in Adabas Link Routines

Addressing Mode Assembly Directives in Adabas Link Routines

All Adabas 8 link routines include `AMODE` and `RMODE` assembly directives. These assembly directives allow the linkage editor to produce warning messages when conflicting `AMODE` or `RMODE` linkage-editor control statements are encountered in the link JCL, JCS, or EXECs.

These assembly directives also serve to document the preferred `AMODE` and `RMODE` for each link routine. It is important to note that in and of themselves, these directives do not alter the actual addressing mode of the link routine during execution.

Re-linking Adabas 8 Link Routines

When re-linking the Adabas 8 link routines with certain `AMODE` and `RMODE` combinations, a warning message may be generated by the linkage editor. This may be safely ignored as long as it pertains to a conflict of `AMODE` or `RMODE` in the ESD record of one or more of the load modules that comprise the link routine, and as long as the resulting module has the proper `AMODE` and `RMODE` attributes for execution with the intended calling application programs.

Care must be taken to ensure that `AMODE(24)` applications will operate properly when invoking the link routine with the attributes chosen when it is re-linked. This is particularly important if the `RMODE(ANY)` attribute is associated with a link routine that will be loaded dynamically but invoked by a program that is `AMODE(24)`. In this case, the link routine should be re-linked `AMODE(31),RMODE(24)` to avoid addressing exception ABENDs because the `AMODE(24)` application cannot correctly invoke the link routine if it resides above the 16-megabyte line.

The Adabas 8 link routines all run `AMODE(31)` after initialization, but they will return to the caller in the caller's `AMODE`.

Note: Under CICS, the version 8 links run `AMODE(31)`, but the Dataloc RDO parameter governs the `AMODE` and `RMODE` of the running CICS transaction.

The batch/TSO non-reentrant link routine, `ADALNK`, has been assembled and linked with `AMODE(31),RMODE(24)`, and that is the recommended configuration to support `AMODE(24)` or

RMODE(24) application programs. It may be re-linked AMODE(31),RMODE(ANY) if desired, but this should only be done if it is certain that all calling programs are AMODE(31).

The ADALNKR batch TSO reentrant link routine has been assembled and link-edited with AMODE(31),RMODE(ANY). If it is loaded by an application that is AMODE(24), it should be relinked AMODE(31),RMODE(24).

The z/OS Com-plete module ADALCO has been assembled and linked AMODE(31),RMODE(ANY). The Com-plete TP monitor ensures proper AMODE switching between AMODE(24) or RMODE(24) programs that invoke ADALCO through the Com-plete Adabas interface routine, TLOPADAB.

All of the V8 CICS link routine modules - ADACICS, ADACICT, ADACICO and ADACIRQ - have been assembled and link-edited AMODE(31),RMODE(ANY). CICS manages the loading of programs and their invocation depending on the DATALOC values associated with their program and transaction definitions.

The Adabas IMS interface link routine ADALNI has been assembled and link-edited AMODE(31),RMODE(ANY). This is the preferred configuration for modern IMS applications, but if there are still AMODE(24) IMS applications executing at your installation, ADALNI may be re-linked AMODE(31),RMODE(24).

ADAUSER AMODE/RMODE Considerations

Software AG recommends that all batch applications invoke Adabas calls through the ADAUSER module. This module is normally link-edited with the application program and it then loads the appropriate link routine as well as ADARUN and ADAIOR/ADAIOS. The source member has the AMODE and RMODE directives coded as AMODE 31, RMODE ANY. This is the most flexible configuration for assembling and linking ADAUSER with the widest variety of application programs. However, if ADAUSER is dynamically loaded, either the RMODE assembler directive should be changed to RMODE 24 before re-assembling it or the ADAUSER module should be re-linked AMODE(31),RMODE(24) to ensure that AMODE 24 application programs may invoke it properly below the 16-megabyte line.

Installing Adabas with IMS/TM under Adabas 8

This section describes installation of the Adabas link routine for the IMS/TM TP monitor with Adabas 8.

IMS requires an Adabas link routine if it is to communicate with Adabas databases. The Adabas Version 8 executable default link routine is delivered in member ADALNI of the AII*vrs*.LOAD library (where *vrs* is the number of the latest Adabas version delivered on the installation medium). If you want to modify this link routine, use member ADALNI8 to do so. ADALNI8 must be linked with a link globals module you prepare and with any link routine exits you require to create the final ADALNI load module that is loaded by the IMS message processing program (MPP) regions

when an application calls them. Members ADALNI and ADALNI8 are provided with some default settings.

This section covers the following topics:

- IMS/TM Link Routines for Adabas 8
- Obtaining the Adabas User ID
- Obtaining the SAF ID
- Installation Procedure under Adabas 8

IMS/TM Link Routines for Adabas 8

These are Adabas 8 link routines for IMS/TM:

- ADALNI is the executable default module for message processing programs (MPPs). If you require no changes to the defaults provided in the link routine, use this module.
- Use ADALNI8 as the base module for message processing programs (MPPs). If you need to tailor ADALNI for your installation, use ADALNI8 to generate an updated ADALNI.
- ADALNK is the batch Adabas link routine for batch message processing (BMP) programs, batch-oriented BMP programs, and batch processing programs (DLIBATCH).

ADALNI and ADALNK use the CSECT name and ENTRY directive ADABAS by default.

The Adabas Version 8 ADALNI and ADALNK are UES-enabled as distributed.

This section describes using ADALNI and ADALNI8 only. For information on using ADALNK, read *General Considerations for Installing Adabas with Batch/TSO*, elsewhere in this guide.

Obtaining the Adabas User ID

The Adabas user ID is obtained at execution time by the ADALNI load module from the LTERM field (first eight bytes) of the IOPCB. The user ID is stored in the Adabas user block field UBUID and will be used for the last eight bytes of the Adabas communication ID.

Obtaining the SAF ID

The SAF ID is supported for use by Adabas SAF Security (ADASAF) if an external security package such as IBM's RACF or CA's ACF2 is present. The SAF ID is obtained at execution time by the ADALNI load module from the user ID field (bytes 33-40) in the IOPCB. To get a valid SAF user ID, SAF sign-on must be active in your IMS installation and the user must have performed an IMS /SIGN command to log onto an IMS terminal.

Installation Procedure under Adabas 8

To modify the default settings and prepare the Adabas 8 link routine for IMS

1 Copy the sample member LNIGBL provided in the Adabas 8 AII*vrs*.SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this guide.

2 Modify the LNIGBL member in the user source library.

Note: The OPSYS parameter must be set to ZOS.

3 Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the Adabas 8 ADA*vrs*.JOBS library. When fully modified, the SET statement in the job should reference the LNIGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the LNIGBL member.

Once modified, submit the ASMGBLS job to assemble and link-edit the link globals module.

A new link globals module (with the name specified by the GBLNAME parameter in LNIGBL) will be generated in the user load library identified in the ASMGBLS job.

4 Copy sample job LNKLNI8 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALNI8 base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLNI8 can be found in the Adabas 8 AII*vrs*.SRCE library.

The module resulting from this job is ADALNI.

5 Place the ADALNI module in a load library available for IMS MPP regions.

The Adabas 8 link routine is prepared.

General Considerations for Installing Adabas with CICS

The macro-level link routine ADALNC is no longer supported for all levels of CICS running under z/OS. These environments must run a current version of Adabas and use the supplied command-level link component.

The Adabas command-level link routine supports the CICS transaction server (CTS) environment.

Notes:

1. The OPID option for the USERID field is not supported under CICS/TS 1.1 and above; therefore, it is not provided with the command-level link routine.
2. The CICS components from Adabas 7.4 or later are required when running with an Adabas 8 SVC.

The following sections describe specific points of Adabas/CICS installation and operation from the CICS perspective:

- Adabas Bridge for VSAM Considerations
- CICS MRO Environment Requirements
- Using CICS Storage Protection
- Sample Resource Definitions

Adabas Bridge for VSAM Considerations

If you are running Adabas Bridge for VSAM 4.2 or 5.1 under CICS, you must run CICS 3.3 or above and the Adabas Version 7.1 or above command-level link routine.

CICS MRO Environment Requirements

If you run the Adabas CICS command-level link routine with the CICS multiple region option (MRO), you must set the `MRO` parameter to "YES" and use the default for the `NETOPT` parameter. In an Adabas 8 installation, these parameters are supplied via the LGBLSET macro (read ***Modifying Source Member Defaults (LGBLSET Macro) in Version 8***, elsewhere in this section).

You can use the LGBLSET `NTGPID` parameter to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when applications that call Adabas span multiple application regions.

Alternatively, you can create a user exit for the link routine that:

- sets UBFLAG1 (byte X'29' in the UB DSECT) to a value of X'08' (UBF1IMSR); and
- places a 4-byte alphanumeric value in the UB field UBIMSID.

This exit is link user exit 1 (LUEXIT1). The exit then allows the Adabas SVC to provide a proper Adabas communication ID in the Adabas command queue element (CQE) even when transactions originate in multiple regions.

Using CICS Storage Protection

The storage protection mechanism (STGPROT) was introduced under CICS/ESA 3.3. Storage protection permits resources to access either CICS or user storage by using the storage protection keys.

- User keys may not overwrite CICS storage, thus affording a degree of protection to CICS.
- CICS keys may read or write either CICS or user key storage, affording the highest degree of access to CICS resources.

Transaction isolation is an extension of the storage protection mechanism. It further protects CICS resources by isolating them in subspaces. This protects user key resources from one another, and protects CICS key resources from the CICS kernel. Transaction isolation can be enabled globally through the CICS TRANISO system initialization (`SIT`) parameter, and for each CICS transaction with the new resource definition `ISOLATE` keyword. Transaction isolation places some restrictions on CICS resources that must be available both during the life of the CICS system and to all transactions running in the CICS system.

In Adabas 8 installations, the CICS link routine always uses a task-related user exit, module ADACICT, so storage isolation is supported by default.

Sample Resource Definitions

Under CICS/TS 1.1 and above for z/OS and VSE, the preferred method for defining and installing CICS programs and transactions is RDO (resource definition online). The CICS documentation no longer recommends the assembly of PPT and PCT entries to define resources.

Modify and use the sample DEFINE statements located in member DEFADA8 as input to the IBM DFHCSDUP utility to define the Adabas CICS command-level components. Consult the appropriate IBM CICS documentation for information on the DFHCSDUP utility. The DEFADA8 member can be found in the Adabas 8 CICS command-level source library (ACI*vrn*.SRCE).

Installing Adabas with CICS under Adabas 8

- Supplied Modules
- Installation Procedure Under Adabas 8

- Preparing DDLINK Input for CICS

Supplied Modules

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with CICS under Adabas 8.

Module	Description
ADACICS	CICS command-level module.
ADACICT	CICS task-related user exit (TRUE) module.

Installation Procedure Under Adabas 8

To install the Adabas 8 CICS link routine components, complete the following steps

1 Copy the Adabas 8 CICS load modules from the Adabas distribution library to a load library that will be in the CICS DFHRPL concatenation (see sample member CPYCICSM in the Adabas 8 ADA*vrn*.JOBS library).

2 Modify the sample CICSGBL member found in the Adabas 8 ACI*vrn*.SRCE library. This member contains sample default installation (LGBLSET) parameter settings. For more information about what to modify in this member, read *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this section.

Note: The OPSYS parameter must be set to ZOS.

3 Save the modified CICSGBL member with a unique name in an appropriate user source library.

4 Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the Adabas 8 ADAvrs.JOBS library. When fully modified, the SET statement in the job should reference the CICSGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the CICSGBL member.

5 Review and run the LNKGCICS member in the ACI*vrn*.SRCE library to link the newly assembled globals table from the previous step with any user or Software AG product exits. (For information about specific Software AG product exits, read the installation documentation for the product.) The LNKGCICS member provides specific instructions. Be sure to link the globals table into a load library that will be made available to CICS in the DFHRPL library concatenation. Note that any user or Software AG link routine exits should be link-edited with this load module.

6 Modify the DEFADA8 member to provide the correct name of the link routine globals default table created in the previous step (Step 4). The default module name is CICSGBL. Tailor this member for any other CICS installation values as required.

Note: The Adabas 8 CICS program names, other than the name of the link globals table, are predefined and cannot be changed (for example, ADACICS, ADACICT, ADACIRQ, and ADACIC0).

7 Run the IBM DFHCSDUP utility to update the CICS CSD file for the desired CICS using the modified DEFADA8 member as input.

8 Modify the CICS PLTPI table to add the entries that will enable and start the Adabas CICS task-related user exits (TRUE). Use member ADAPLTXX from the Adabas 8 ACI*vrn*.SRCE library as a sample for enabling and starting a legacy Adabas TRUE and the new Version 8 TRUE in the second phase of the PLT.

9 Assemble and link the modified PLTPI table into a library that will be available to the desired CICS region.

10 Modify the CICS startup JCL to include the DDLINK DD statement that provides the name of the link globals table prepared in Step 4. For more information on doing this, read ***Preparing DDLINK Input for CICS***, elsewhere in this section.

11 Start the CICS and note any messages relating to the installation of the Adabas TRUE modules that appear on the console.

Preparing DDLINK Input for CICS

Operation of the Adabas 8 CICS link routines may be tailored for each CICS address space by assembling and linking a link globals defaults table and making that table available to CICS at execution. The table may have any legal load module name that is acceptable to CICS and that does not conflict with existing load module names used in the CICS region.

The globals table must be defined to CICS as a program. Review the provided sample DEFADA8 member found in the Adabas 8 ACI*vrn*.SRCE library to see a definition of the sample called LNKGBLS. The DEFADA8 member should be modified as necessary and provided as input to the DFHCSDUP utility to define the Adabas 8 components in the CICS CSD. This process is also described as part of the installation procedure under Adabas 8. Consult the IBM CICS documentation for information on the DFHCSDUP utility.

Each link routine globals table for a CICS region may have a unique name. The Adabas 8 CICS link routines are provided this name through an external transient data queue. The queue-name is ADAI and its definition is also provided in the DEFADA8 member.

When the Adabas 8 task-related user exit (TRUE) is enabled via the ADACIC0 program, either at CICS startup or with the ADA0 transaction, the ADACIRQ module is invoked and reads the ADAI transient data queue. The data read is provided in a file, or a partitioned data set member, or as system input through the DDLINK DD statement. The CICS JCL must be modified to provide this DD statement or the default link globals table name "LNKGBLS" will be used. If no link globals table is located, the Adabas 8 TRUE will not be enabled and started.

The format of the input data to be read is:

Content	Description
ADALNK	This keyword must appear in columns 1 through 6.
space	A blank space must appear in column 7.
LGTNAME= or LGT=	The keyword LGTNAME or LGT, followed by an equals sign (=) must appear after the space in column 7, starting in column 8.
module-name	The module name of the prepared link globals table must appear after the equal sign that follows the LGTNAME or LGT keyword.

For example, the following might be added to the CICS JCL:

```
//DDLINK DD  *
ADALNK LGTNAME=LNKGBLS
/*
```

In this example, the link default globals table named "LNKGBLS" must be prepared, assembled, link-edited, and defined to this CICS.

Installing Adabas with Com-plete under Adabas 8

Certain Adabas parameters are required by Com-plete, Software AG's TP monitor, when installing Adabas. For more information, see the *Com-plete System Programmer's* manual.

Software AG's TP monitor, Com-plete requires an Adabas link routine if it is to communicate with Adabas databases, use Software AG's Entire Net-Work product, or use products like Entire System Server running under Com-plete. At this time, Com-plete does not support a mixed Adabas 7 and Adabas 8 link routine environment; thus Com-plete must be run with either an Adabas 7 link routine or an Adabas 8 link routine.

The Adabas Version 8 link routine is delivered in member ADALCO of the Adabas 8 z/OS load library. This member must be linked with a link globals module you prepare and with any link routine exits you require to create the final ADALCO load module that is loaded by Com-plete when Com-plete is initialized. The final ADALCO load module and any exits linked with it must be reentrant.

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with Com-plete under Adabas 8.

Module	Description
ADALCO8	Base module
ADALCO	Executable default module

To prepare the Adabas 8 link routine

1 Copy sample member LCOGBL provided in the Adabas 8 ADA*vrs*.SRCE library to any appropriate user source library where it can be modified (where *vrs* is the number of the latest Adabas version delivered on the installation medium). LCOGBL is a module containing LGBLSET parameters that are used to create default settings for command-level link components. A complete description of LGBLSET parameters can be found in ***Modifying Source Member Defaults (LGBLSET Macro) in Version 8***, elsewhere in this guide.

2 Modify the LCOGBL member in the user source library.

At a minimum supply values for the following LGBLSET parameters in LCOGBL:

Parameter	Specify...
LOGID	The default database or target ID. This should be a numeric value between "1" and "65535". The default value is "1". **Note:** Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.
SVCNO	The default Adabas SVC number. For z/OS, this number should be between "200" and "255". **Note:** Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.
OPSYS	The three-character abbreviation for the operating system under which Com-plete executes. Valid values include "ZOS" and "VSE". **Note:** The OPSYS parameter must be set to ZOS.
TPMON	COM. This keyword specifies the three-character TP monitor abbreviation. For Com-plete, this value should be "COM".
RENT	YES. This keyword indicates whether or not the module is serially reentrant. For Com-plete, this value should be "YES".
GEN	CSECT. This keyword indicates whether a CSECT or DSECT is generated. CSECT must be specified so an object module is generated that can be linked as the link routine globals load module.
UES	Whether Adabas Universal Encoding Support (UES) should be enabled. The default is YES.

Parameter	Specify...
exit parameters	Whether any other exits are to be active, and in the case of user exits you provide, specify the user exit module names. Specify this information in other parameters of LGBLCOM, as described in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this guide.

3 Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the Adabas 8 ADA*vrs*.JOBS library. When fully modified, the SET statement in the job should reference the LCOGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the LCOGBL member.

Once modified, submit the ASMGBLS job to assemble and link-edit the link globals module.

A new link globals module (with the name specified by the GBLNAME parameter in LCOGBL) will be generated in the user load library identified in the ASMGBLS job.

4 Copy sample job LNKLCO8 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALCO8 base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLCO8 can be found in the Adabas 8 ADA*vrs*.SRCE library.

The module resulting from this job is called ADALCO.

5 Place the ADALCO module in a load library available in the job step that will start Com-plete.

The Adabas 8 link routine is prepared.

General Considerations for Installing Adabas with Batch/TSO

When installing Adabas 8 on TSO systems, Adabas-TSO communication is provided by the batch link routines ADALNK8 (non-reentrant) and ADALNKR8 (reentrant).

The Adabas Version 8.1 ADALNK routines are UES-enabled as distributed.

However, it is important to note that user programs linked with ADAUSER also load ADARUN. ADARUN, in turn, loads other modules.

To start a user program linked with ADAUSER, the following modules must all be available from the defined load libraries for that specific TSO user at execution time:

```
ADAIOR ADAMLF
ADAIOS ADAPRF
ADALNK ADARUN
```

This section covers the following topics:

- Non-reentrant ADALNK Batch Routine Operation
- ADALNKR: Reentrant Batch Link Routine

Non-reentrant ADALNK Batch Routine Operation

The ADALNK module in the Adabas 8 load library operates in an Adabas 7-compatible manner when the following conditions are met:

- The calling application must be linked with ADAUSER. If the calling application is not linked with ADAUSER, the ADALNK will not work.
- The ADARUN module from the most recent Adabas 8 load library must be used.
- The database ID and Adabas SVC number must be provided as input through DD statements. Otherwise, the values in the link globals table will override these values.

If all three of these conditions are met, the default database ID and Adabas SVC number will be overridden by the values provided in the DD statement input and passed to the link routine by ADARUN.

Operating in this fashion requires the fewest changes on the part of your data base administrator (DBA) and application programmer. This is also the recommended mode of operation when executing Adabas utilities.

ADALNKR: Reentrant Batch Link Routine

Several Software AG products require the use of a reentrant batch link routine and the ADALNKR load module is provided in the Adabas load library to support them. The Adabas 8 ADALNKR source module is not provided.

You can change default values for these reentrant batch link routines. For more information, read one of the following sections, elsewhere in this section:

- ***Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules***
- ***Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules***

Software AG recommends that batch application programs be linked with the ADAUSER module, not ADALNK or ADALNKR. The ADAUSER load module is not reentrant, but the reentrant link routine module may be linked with it as long as the application program conforms to the calling requirements described in *Adabas 8 Batch/TSO Reentrant Link Routine (ADALNKR) Calling Requirements*(in *Adabas Operations Manual*) and the PROG=RENTUSER ADARUN parameter is provided in DDCARD input instead of the keyword parameter PROG=USER.

When using the latest Adabas 8 ADALNKR module to obtain reentrant operation under batch or TSO, you must prepare the ADALNKR module in advance. It must be linked with a customized link globals table that provides defaults for the database ID, Adabas SVC number, and other requirements. Any reentrant exits should also be linked with it as required.

Installing Adabas with Batch/TSO under Adabas 8

When installing Adabas 8 on TSO systems, the standard Adabas 8 batch link routine (ADALNK) provides Adabas/TSO communication (SMA job number I056).

This section covers the following topics:

- Supplied Modules
- Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules
- Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules

Supplied Modules

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with batch/TSO under Adabas 8.

Module	Description
ADALNK8	Base module
ADALNKR8	Base reentrant module
ADALNK	Executable default module
ADALNKR	Executable default reentrant module

Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules

You can change default values for various link routine parameters used by the Adabas 8 ADALNK and ADALNKR modules.

To change default values, complete the following steps

1 Copy the sample member LNKGBLS (for non-reentrant links) or LNKRGBL (for reentrant links) members provided in the Adabas 8 ADA*vrs* (where *vrs* is the number of the latest Adabas version delivered on the installation medium).SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this guide.

2 Modify the LNKGBLS or LNKRGBL member in the user source library. Provide values for the LOGID, SVC, and other keywords to suit your installation requirements.

Note: The OPSYS parameter must be set to ZOS.

3 Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the Adabas 8 ADA*vrs*.JOBS library. When fully modified, the SET statement in the job should reference the LNKGBLS or LNKRGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBL-NAME parameter in the LNKGBLS or LNKRGBL member.

Once modified, submit the ASMGBLS job to assemble and link-edit the link globals module.

A new link globals module (with the name specified by the GBLNAME parameter in the LNKGBLS or LNKRGBL member) will be generated in the user load library identified in the ASMGBLS job.

4 Copy sample job LNKLNK8 or LNKLNKR8 (reentrant) to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the appropriate ADALNK8 or ADALNKR8 (reentrant) base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from the job to an appropriate user load library. LNKLNK8 and LNKLNKR8 can be found in the Adabas 8 ADA*vrs*.SRCE library.

The module resulting from this job is called ADALNK or ADALNKR (as appropriate).

5 Tailor the ADARUN DDCARD input for the job steps that will use the Adabas 8 batch/TSO link routines. The DDCARD input should include the following updates:

- Specify the ADARUN PROG=USER parameter for a non-reentrant link routine, or specify ADARUN PROG=RENTUSER to use a reentrant link routine in the job step.

6 Make sure the appropriate load libraries are made available to the job step. These may be STEPLIB, TASKLIB, JOBLIB, or, for reentrant modules, the LPA or LINKLIB.

Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules

You can change default values for various link routine parameters used by the Adabas 8 ADALNK and ADALNKR modules.

Changes to some default values for the Adabas 8 batch/TSO link routines, ADALNK and ADALNKR, may occur with a zap to either the ADALNK or ADALNKR module. This includes the default values for the database ID and the Adabas SVC number. All other default values should be set using the link globals table, as described in *Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules*, earlier in this section.

Use the following IMASPZAP control statements to change default values in ADALNK or ADALNKR (as appropriate):

```
NAME ADALNK ADALNK8
VER 0080 0001                Default DBID
REP 0080 ####                Site-specific DBID
VER 0084 0AF9                Default Adabas SVC number
REP 0084 0A##                Site-specific Adabas SVC number
*
NAME ADALNKR ADALNKR8
VER 0080 0001                Default DBID
REP 0080 ####                Site-specific DBID
VER 0084 0AF9                Default Adabas SVC number
REP 0084 0A##                Site-specific Adabas SVC number
```

Modifying Source Member Defaults (LGBLSET Macro) in Version 8

The Adabas 8 LGBLSET macro is used to set default installation values for the Adabas link routines. It is used to prepare an object module which may either be link-edited with the Adabas 8 link routines or provided to the link routines in the job step where they are run. Your Adabas libraries include sample members provided to support the various teleprocessing (TP) monitors in each environment. Each of these sample members may be copied to an appropriate library and modified to provide the necessary customization required for the link routine that is intended to run in a given environment.

The LGBLSET parameter options with their default values (underlined) are described in the rest of this section:

- ADL: Adabas Bridge for DL/I Support
- AVB: Adabas Bridge for VSAM Support
- CITSNM: Adabas CICS TS Queue Name
- COR: SYSCOR Exit Support
- DBSVCTN: DBID/SVC Routing Table
- DYNDBSVC: DBID/SVC Routing Table
- ENTPT: Name of the Adabas CICS Command-Level Link Routine
- GBLNAME: Name of Link Globals Module
- GEN: Generate CSECT or DSECT
- IDTNAME: BS2000/OSD IDT Common Memory Name
- IDTUGRP: BS2000/OSD Memory Pool User Bound
- LOGID: Default Logical Database ID
- LUEXIT1A: Length of LUEXIT1
- LUEXIT2A: Length of LUEXIT2
- LUINFO: Length of User Data passed to Adabas LUEXIT1 and LUEXIT2
- LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2
- LX1NAME: User Exit 1 Module Name

- LX2NAME: User Exit 2 Module Name
- MRO: Multiple Region Option
- NETOPT: Method Used to Create User ID
- NTGPID: Natural Group ID
- NUBS: Number of User Blocks Created By CICS Link Routine
- OPSYS: Operating System
- PARMTYP: Area for Adabas Parameter List
- PRE: DSECT Data Prefix
- PURGE: Purge Transaction
- RENT: Reentrant Module Flag
- RETRYX: Retry Command Exit Flag
- REVIEW: Adabas Review Support
- RMI: Resource Manager Interface
- RTXNAME: Command Retry Exit Name
- SAF: Adabas Security Interface Flag
- SAP: SAP Application Support
- SAPSTR: SAP ID String
- SVCNO: Adabas SVC number
- TPMON: Operating Environment
- TRUENM: CICS TRUE Name
- UBPLOC: User Block Pool Allocation
- UES: Universal Encoding Support
- USERX1: User Exit 1 Flag
- USERX2: User Exit 2 Flag
- XWAIT: XWAIT Setting for CICS

ADL: Adabas Bridge for DL/I Support

Parameter	Description	Syntax
ADL	Indicates whether or not the Consistency Interface of Software AG's Adabas Bridge for DL/I is to be supported by this command-level link routine. ■ ADL=YES: Adabas Bridge for DL/I Consistency Interface is to be supported. ■ ADL=NO: Adabas Bridge for DL/I Consistency Interface is *not* to be supported.	ADL={NO|YES}

AVB: Adabas Bridge for VSAM Support

Parameter	Description	Syntax
AVB	Indicates whether or not Software AG's Adabas Bridge for VSAM is to be supported by this command-level link routine. ■ AVB=YES: Adabas Bridge for VSAM is to be supported. ■ AVB=NO: Adabas Bridge for VSAM is *not* to be supported.	AVB={NO|YES}

CITSNM: Adabas CICS TS Queue Name

Parameter	Description	Syntax
CITSNM	Specifies the 16-byte string that represents the CICS TS queue name for Adabas. The default is "ADACICS".	CITSNM={ADACICS|*qname*}

COR: SYSCOR Exit Support

Parameter	Description	Syntax
COR	Indicates whether or not Adabas System Coordinator (SYSCOR), Adabas Transaction Manager, and Adabas Fastpath exits are installed and active. ■ COR=YES: The exits are installed and active. ■ COR=NO: The exits are *not* installed and active.	COR={NO|YES}

DBSVCTN: DBID/SVC Routing Table

Parameter	Description	Syntax
DBSVCTN	Provides the name of the DBID/SVC routing table that should be used by the link routine during its execution, if any. The routing table name must conform to names for z/OS standard load modules. It is used by a z/OS LOAD macro/SVC during batch, TSO, or IMS operation or by an EXEC CICS LOAD PROGRAM command during CICS operation. If the load module listed is not found, or if it is found to contain invalid header information, user abend U657 is issued in batch, TSO, or IMS environments. If the load module is not defined to CICS or not found in the CICS DFHRPL concatenation, the Adabas CICS link routine environment is not initialized.	DBSVCTN={*name*|ADASVCTB}

Parameter	Description	Syntax
	Note: If the DYNDBSVC parameter is set to NO, this parameter setting is ignored.	

DYNDBSVC: DBID/SVC Routing Table

Parameter	Description	Syntax
DYNDBSVC	Indicates whether Adabas SVC routing by database ID should be enabled for the link routine. DYNDBSVC=YES enables Adabas SVC routing by database ID; DYNDBSVC disables it. The default is NO.	DYNDBSVC={YES|NO}

ENTPT: Name of the Adabas CICS Command-Level Link Routine

Parameter	Description	Syntax
ENTPT	The name given to the Adabas CICS command-level link routine. This name is used in EXEC CICS LINK commands to invoke Adabas services from CICS application programs. See also notes 1 and 2 in the installation procedure.	ENTPT={ADACICS|*name*}

GBLNAME: Name of Link Globals Module

Parameter	Description	Syntax
GBLNAME	The name of the link globals module.	GBLNAME={LNKGBLS|*name*}

GEN: Generate CSECT or DSECT

Parameter	Description	Syntax
GEN	Indicates whether a CSECT or DSECT is generated.	GEN={CSECT|DSECT}

IDTNAME: BS2000/OSD IDT Common Memory Name

Parameter	Description	Syntax
IDTNAME	The common memory pool name of the BS2000/OSD IDT.	IDTNAME=*name*

IDTUGRP: BS2000/OSD Memory Pool User Bound

Parameter	Description	Syntax
IDTUGRP	Indicates whether the common memory pool is user bound (BS2000)	IDTUGRP={NO|YES}

LOGID: Default Logical Database ID

Parameter	Description	Syntax
LOGID	The value of the default target database ID. Valid ID numbers are 1-65535. The default is "1".	LOGID={*nnn* | 1}

LUEXIT1A: Length of LUEXIT1

Parameter	Description	Syntax
LUEXIT1A	The length of the work area for link user exit 1. Valid values are numbers from zero (0) through 32,767. The default is "0".	LUEXIT1A={*nnn*|0}

LUEXIT2A: Length of LUEXIT2

Parameter	Description	Syntax
LUEXIT2A	The length of the work area for link user exit 2. Valid values are numbers from zero (0) through 32,767.The default is "0".	LUEXIT2A={*nnn*|0}

LUINFO: Length of User Data passed to Adabas LUEXIT1 and LUEXIT2

Parameter	Description	Syntax
LUINFO	The length of the user data to be passed to target user exit 4. Valid values are numbers from zero (0) through 32,767. If LUINFO is not specified, the default is zero (no user save area is passed).	LUINFO={0|*length*}

LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2

Parameter	Description	Syntax
LUSAVE	The size of the user save area to be used by Adabas user exits LUEXIT1 and LUEXIT2. Valid values range from zero (0) through 256. The default is "72". If LUSAVE is not specified, the default is zero (no user data is passed).	LUSAVE={72|*size*}

LX1NAME: User Exit 1 Module Name

Parameter	Description	Syntax
LX1NAME	The name of the link user exit 1 module	LX1NAME={LUEXIT1|*name*}

LX2NAME: User Exit 2 Module Name

Parameter	Description	Syntax
LX2NAME	The name of the link user exit 2 module	LX2NAME={LUEXIT2|*name*}

MRO: Multiple Region Option

Parameter	Description	Syntax
MRO	Indicates whether or not the CICS multiple region option (MRO) support is required. If you run the CICS command-level link with the CICS MRO, set this to MRO=YES; otherwise, use the default value MRO=NO. If MRO=YES, NETOPT must be set to NETOPT=NO (the default) to prevent non-unique LU names from multiple application regions. If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.	MRO={NO|YES}

NETOPT: Method Used to Create User ID

Parameter	Description	Syntax
NETOPT	If NETOPT=YES is specified, an 8-byte user ID will be constructed from the VTAM LU name. If NETOPT=NO is specified, the user ID is created from the constant CICS plus the four-byte CICS terminal ID (TCTTETI) for terminal tasks. For non-terminal tasks, the user ID comprises the constant CIC plus the CICS task number. If you run with the CICS multiple region option (MRO), you must use the default value for this option. If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.	NETOPT={NO|YES}

NTGPID: Natural Group ID

Parameter	Description	Syntax
NTGPID	Specifies a four-byte Natural group ID as required for unique Adabas user ID generation in the CICSplex environment with Natural Version 2.2.8 and above. The value is associated with all users who call the Adabas command-level link routine assembled with the specified value. There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner. Any four-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICSplex, or z/OS image. If more than one NTGPID is required (for example, both test and production Natural 2.2.8), more than one Adabas command-level link routine with associated TRUE must be generated. If you run with the CICS multiple region option (MRO), you may use NTGPID to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when multiple application regions call Adabas.	NTGPID=*4-byte-value*

NUBS: Number of User Blocks Created By CICS Link Routine

Parameter	Description	Syntax
NUBS	The number of user blocks (UBs) to be created in the user block pool by the CICS link routine. The number of blocks must be large enough to handle the maximum possible number of concurrent Adabas requests. **Note:** The Adabas 6.2 and above command-level link routine obtains storage for the user blocks (the UB pool) above the 16-megabyte line.	NUBS={100|*blocks*}

OPSYS: Operating System

Parameter	Description	Syntax
OPSYS	The operating system in use.	OPSYS={ZOS|VSE|CMS|BS2}

PARMTYP: Area for Adabas Parameter List

Parameter	Description	Syntax
PARMTYP	The CICS area which is to contain the Adabas parameter list. "TWA" picks up the parameter list in the first six fullwords of the transaction work area (TWA). When PARMTYP=COM, the Adabas parameters are supplied in the CICS COMMAREA provided by the calling program with the EXEC CICS LINK command. The COMMAREA list for an ACB call must be at least 32 bytes long and begin with the label "ADABAS52". The COMMAREA list for an ACBX call must be at least 24 bytes long and begin with the label "ADABAS8X". In addition, the last ABD in the COMMAREA list for an ACBX call must be indicated by setting the VL-bit -- in other words, the high bit in the address must be on (X'80'). PARMTYP=ALL (the default) uses both the COMMAREA and TWA to pass the Adabas parameters; in this case, the COMMAREA is checked first. We do not recommend that you attempt to map the CICS TWA to the Adabas 8 ACBX direct call. This is because the TWA is of finite size per transaction and because the TWA in not available at CICS startup. We therefore recommend that CICS programs using the Adabas 8 CICS link routines use the COMMAREA only for passing data.	PARMTYP={ALL|COM ↩ |TWA}

PRE: DSECT Data Prefix

Parameter	Description	Syntax
PRE	The two-byte string to be used as the DSECT data prefix. The default is "LG".	PRE={LG|*prefix*}

PURGE: Purge Transaction

Parameter	Description	Syntax
PURGE	The PURGE parameter is used when assembling with CICS 3.2 or above. If PURGE=YES is specified, the CICS WAIT EXTERNAL will contain PURGEABLE as one of its parameters, allowing the transaction to be purged by CICS if the DTIMOUT value is exceeded and PURGE is specified. If PURGE=NO (the default) is specified, the NONPURGEABLE option is generated.	PURGE={NO|YES}

RENT: Reentrant Module Flag

Parameter	Description	Syntax
RENT	Indicates whether the globals module is reentrant.	RENT={NO|YES}

RETRYX: Retry Command Exit Flag

Parameter	Description	Syntax
RETRYX	Indicates whether the retry command exit is active.	RETRYX={NO|YES}

REVIEW: Adabas Review Support

Parameter	Description	Syntax
REVIEW	Indicates whether or not Software AG's Review performance monitor is installed and active.	REVIEW={NO|YES}

RMI: Resource Manager Interface

Parameter	Description	Syntax
RMI	The RMI parameter is used to indicate whether or not the CICS Resource Manager Interface is in use. If RMI=YES is specified, the Adabas task-related user exit (TRUE) will be executed as a resource manager (RM) using the CICS Resource Manager Interface (RMI). RMI=YES is valid only when the Adabas Transaction Manager is installed, enabled, and available to users executing in the CICS environment. Consult the Adabas Transaction Manager documentation for additional instructions related to the installation of the Adabas TRUE.	RMI={NO|YES}

RTXNAME: Command Retry Exit Name

Parameter	Description	Syntax
RTXNAME	The name of the command retry exit module.	RTXNAME={LUEXRTR|*name*}

SAF: Adabas Security Interface Flag

Parameter	Description	Syntax
SAF	Indicates whether Software AG's Adabas SAF Security support is required.	SAF={NO|YES}

SAP: SAP Application Support

Parameter	Description	Syntax
SAP	Indicates whether or not SAP user ID generation is supported. If SAP=YES is specified, the program will detect a SAP initialization call and set the user ID for SAP applications from the constant provided on the initialization call, plus the field ACBADD2. For more information, refer to the supplementary information provided to customers using the SAP application system.	SAP={NO|YES}

SAPSTR: SAP ID String

Parameter	Description	Syntax
SAPSTR	The four-byte SAP ID string to use.	SAPSTR={'SAP*'|*string*}

SVCNO: Adabas SVC number

Parameter	Description	Syntax
SVCNO	The value of the Adabas SVC number. On z/OS systems, valid values range from 200-255 and the default is "249". On z/VSE systems, valid values range from 32-128 and the default is "45".	SVCNO=*nnn*

TPMON: Operating Environment

Parameter	Description	Syntax
TPMON	The TP monitor operating environment. Valid values should be specified as follows: ■ Specify "BAT" to use batch. ■ Specify "CICS" to use CICS. ■ Specify "COM" to use Com-plete. ■ Specify "IMS" to use IMS.	TPMON={BAT|CICS|COM|IMS}

Parameter	Description	Syntax
	■ Specify "TSO" to use TSO. ■ Specify "UTM" to use UTM. **Caution:** Be sure to specify a TP monitor operating environment that is supported on the operating system you selected in the OPSYS parameter.	

TRUENM: CICS TRUE Name

Parameter	Description	Syntax
TRUENM	Specifies the module name of the Adabas CICS task-related user exit (TRUE). The default is ADACICT.	TRUENM={ADACICT|*name*}

UBPLOC: User Block Pool Allocation

Parameter	Description	Syntax
UBPLOC	Specifies whether the user block (UB) pool is to be obtained above (the default) or below the 16-megabyte line in CICS. The ECB used by the EXEC CICS WAIT WAITCICS or the EXEC CICS WAIT EXTERNAL is included in the UB pool. The UBPLOC=BELOW setting supports versions of CICS that do not allow ECBs above the 16-megabyte line; that is, CICS/ESA 3.2 or below. Refer to the IBM manual *CICS Application Programming Reference* for more information.	UBPLOC={ABOVE|BELOW}

UES: Universal Encoding Support

Parameter	Description	Syntax
UES	Indicates whether or not Universal Encoding Support (UES) is required.	UES={NO|YES}

USERX1: User Exit 1 Flag

Parameter	Description	Syntax
USERX1	Indicates whether or not user exit 1 is active.	USERX1={NO|YES}

USERX2: User Exit 2 Flag

Parameter	Description	Syntax
USERX2	Indicates whether or not user exit 2 is active.	USERX2={NO|YES}

XWAIT: XWAIT Setting for CICS

Parameter	Description	Syntax
XWAIT	Indicates whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be executed by the Adabas 8 task-related user exit (TRUE). XWAIT=YES is the default. The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/TS 1.1 and above. The CICS WAITCICS statement (XWAIT=NO) is provided for use with CICS/MVS 2.1.2 and for CICS/VSE 2.1 through 2.3. It may also be used for CICS/TS 1.1 and above, but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments. **Note:** If XWAIT=NO is specified for use under CICS/ESA 3.3, IBM APAR PN39579 must be applied to the CICS/ESA 3.3 system. For CICS/TS 1.1 and above, this APAR is not required.	XWAIT={NO|YES}

Notes:

1. If XWAIT=NO is specified, the ADACICT (Adabas 8 TRUE) module issues an EXEC CICS WAIT-CICS command instead of the EXEC CICS WAIT EVENT command. XWAIT=YES conforms with recommended IBM usage of the WAIT and ECB lists in a high-transaction volume CICS system with CICS/TS Version 1.1 and above.
2. All EXEC CICS commands are processed by the CICS preprocessor; the LGBLSET parameters cause the subsequent assembly step to skip some of the statements.

XWAIT Posting Mechanisms

CICS WAITCICS (XWAIT=NO) can support a soft post of the specified ECB. This has the disadvantage of becoming a low priority dispatchable unit of work in a CICS environment, since the hand-postable work is not processed by CICS on every work cycle.

EXEC CICS WAIT EXTERNAL (XWAIT=YES), on the other hand, allows CICS to make use of its special post exit code, and will always be checked and processed (if posted) on every CICS work cycle.

For more details on the differences between the various CICS WAIT commands and their relationship to hard and soft posting mechanisms, consult the IBM *CICS Application Programming Reference Guide* and the texts accompanying IBM APAR PN39579 or "Item RTA000043874" on the IBM InfoLink service.

XWAIT and the Adabas SVC / Router

The Adabas SVC is fully compatible with the XWAIT=YES setting. The SVC performs the necessary hard post for Adabas callers under CICS using the Adabas command-level link routine. The same SVC performs a soft post for batch callers where the hard post is not required.